VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Analýza jízdních pruhů pomocí konvolučních neuronových sítí

Road Lane Detection Using Convolutional Neural Network

Adam Maják

Diplomová práce

Vedoucí práce: Ing. Radovan Fusek, Ph.D.

Ostrava, 2021

Abstrakt

Cieľom tejto diplomovej práce je analýza jazdných pruhov. Práca je rozdelená na viac samostatných častí. V úvode sú rozoberané základné charakteristiky jazdných situácií. Ďalšia kapitola rozoberá prístupy a hotové riešenia, s ktorými je možné sa stretnúť. V posledných kapitolách je práca smerovaná k vytvoreniu aplikácie, ktorá využíva časti postupov jednotlivých riešení a experimentálnemu overeniu na vytvorených dátach z lokálneho prostredia.

Kľúčové slová

detekcia obrazu; jazdné pruhy; neurónové siete; polohovanie kamery; Tensorflow; OpenCV; dataset

Abstract

The aim of this diploma thesis is the analysis of lanes. The work is divided into several separate parts. In the introduction, the basic characteristics of driving situations are discussed. The next chapter discusses the approaches and ready-made solutions that can be encountered. In the last chapters, the work is aimed at creating an application that uses parts of the procedures of individual solutions and experimental verification of the created data from the local environment.

Keywords

image detection; lanes; neural networks; camera positioning; Tensorflow; OpenCV; dataset

Poďakovanie

Rád by som poďakoval môjmu vedúcemu práce Ing. Radovanovi Fusekovi, Ph.D. za pomoc a ochotu pri vypracovaní diplomovej práce.

Obsah

Zoznam použitých skratiek a symbolov

3D	–	3 dimension
ADAS	–	Advanced driver-assistance systems
ASCII	–	American Standard Code for Information Interchange
AVG	–	Average
AVV	–	Autonómne vedené vozidlo
CIE	–	International Commission on Illumination
CNN	–	Konvolučná neurónová sieť
CPU	–	Central processing unit
FPS	–	Frames pre second
GB	–	Gigabyte
GHz	–	Gigahertz
GPU	–	Graphics processing unit
LSTM	–	Long short-term memory
MUK	–	Mimoúrovňová križovatka
NMS	–	Non-maximum Suppresion
PCA	–	Principal component analysis
RAM	–	Random Access Memory
ReLU	–	Rectified Linear Units
RIO	–	Región záujmu
RGB	–	Red-Green-Blue
Tanh	–	Hyberbolický tangens
TF	–	Tensorflow
TPU	–	Tensor processing unit

Zoznam obrázkov

Zoznam tabuliek

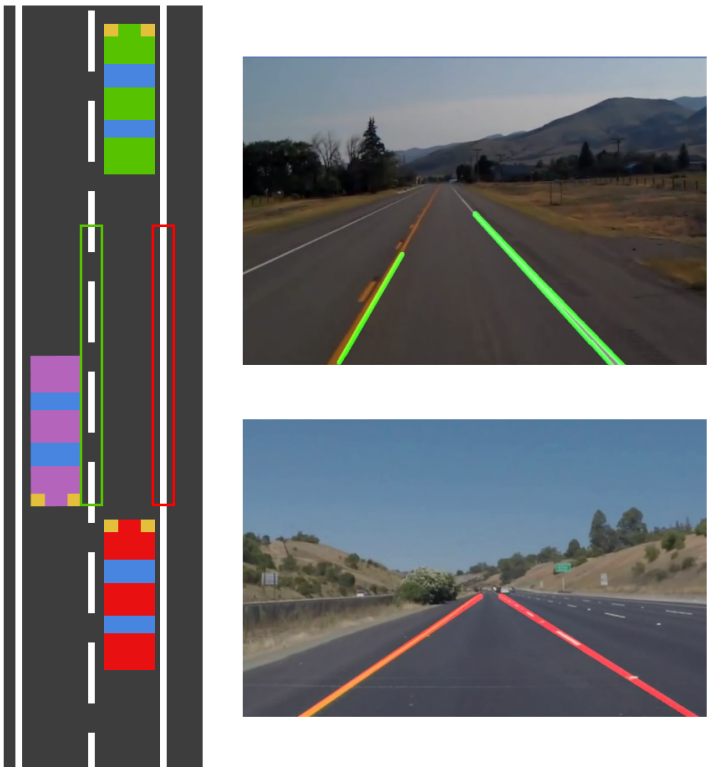
Zoznam výpisov zdrojového kódu

Kapitola 1

Úvod

Dnešná doba napreduje míľovými krokmi. Čím ďalej ľudstvo je, tým viac času trávi cestovaním. Ako však tento čas rozumne využiť? Sedieť za volantom a pozerať ako ubiehajú kilometre, nie je efektívny spôsob trávenia času. Autá, ktoré by jazdili úplne samé sú hudbou blízkej budúcnosti. Avšak každý takýto vynález v sebe zahŕňa veľké množstvo poznatkov, postupov a metód, ktoré si muselo ľudstvo osvojiť a vymyslieť. Automobil, ktorý by bol schopný takto jazdiť musí spĺňať niekoľko predpokladov a musí byť schopný vykonávať niekoľko funkcií ako sú napríklad, brzdenie pred prekážkou, detekcia prekážky alebo držanie sa v jazdnom pruhu. Dnešní výrobcovia automobilov už pomaly uvažujú týmto smerom a začínajú sa orientovať na trhu autonómych vozidiel, ktoré spĺňajú tieto parametre. Hlavný dôraz sa musí klásť na bezpečnosť, aby neboli ohrozené ľudské životy, ktoré sú na palube daného vozidla. Ďalším aspektom je rýchlosť alebo včasná detekcia a vyhnutie sa prekážke alebo správne a rýchle detekovanie jazdných pruhov. Táto detekcia musí byť rýchla a bezchybná. Už aj v tejto chvíli je na trhu množstvo riešení od rôznych autorov či firiem. Niektoré v sebe ukrývajú len malú funkcionalitu, iné sú komplexné a ich využitie nesie na trhu úspech. Taktiež využívajú rôzne programovacie jazyky a knižnice. Líšia sa aj hardvérovou výbavou ako sú, napríklad kamery, či senzory.

Táto diplomová práca je zameraná na poslednú zo spomínaných funkcií a to detekcii jazdných pruhov. Diplomová práca na úvod preberá jednotlivé postupy riešenia tohoto problému, ako je detekcia objektov v obraze bez učenia alebo postup, pri ktorom sa využívajú konvolučné neurónové siete a s tým spojené aj použitie známych knižníc, napríklad Tensorflow alebo Keras.



Obr. 1.1: Detekcia jazndého pruhu

Ako je vidieť na obr. 1.1 zvolený detektor v premávke musí byť schopný vyznačiť čiary jazdného pruhu, v ktorých sa automobil pohybuje. Rôzne typy detektorov používajú na označovanie čiar rôzne techniky. Tieto postupy sú preberané nižšie v kapitole 3. V neskorších kapitolách sa diplomová práca venuje implementácií vybratých riešení a zavedení daných riešení do praxe. Za účelom testovania alebo vylepšenia jestvujúcich riešení bol vytvorený vlastný dataset. Dataset bol vytvorený v lokálnom prostredí pre lepšie adaptovanie sa na lokálne cestné situácie. Konkrétne je analyzovaná cestná infraštruktúra v meste Ostrava a Žilina a to cestné ťahy a cesty I/647 Marianské Hory, cesta Sokolovská škola, ale tiež rýchlostná cesta D3 Svrčinovec-Skalité polovičný profil, kde boli natáčané nočné zábery. Ďalšou cestou je cesta I/11 smerom do Žiliny a taktiež mestský žilinský okruh. Ako kamera je zvolená kamera, ktorú obsahuje mobilný telefón Samsung Galaxy S8 a to pri rozlíšení 1920x1080 pixelov. Ďalšou kamerou je kamera z mobilného telefónu Samsung Galaxy S10 taktiež v rozlíšení Full HD, kde je použitá aj stabilizácia obrazu pre vybrané úseky. Po implementácií je zvolené riešenie otestované, celý postup testovania je zdokumentovaný. V kapitole sa hodnotí celková efektivita zvoleného riešenia, ale aj úspešnosť detekcie na jednotlivých cestných úsekoch. Výsledky sú zobrazené v prehľadných tabuľkách, čo pomôže lepšej vizualizácii.

Kapitola 2

Základné parametre analýzy jazdných pruhov

Súčasné úsilie mnohých spoločností zaoberajúcich sa vývojom automobilov venuje vývoju vozidiel s automatickým riadením veľkú pozornosť. Predpokladá sa, že budúca automobilová doprava pozostávajúca z vozidiel s automatickým riadením by mala výrazne zvýšiť kapacitu diaľnice a predovšetkým eliminovať počet dopravných nehôd spôsobených príčinami subjektívne súvisiacimi s vodičom, ako sú opitosť, únava a nesprávna jazda [1]. Inteligentné vozidlá môžu do určitej miery eliminovať tieto ľudské faktory. Na dosiahnutie automatickej jazdy je nevyhnutnou vlastnosťou rozlišovať rozdiel medzi cestnou a necestnou dopravou na základe kamerových vstupov. Štandardné techniky počítačového videnia dokážu skoro bez problémov analyzovať cesty, ktoré majú jasné hranice okrajov. Problém nastáva pri analýze, kde je povrch vozovky zle vyznačený [2]. Najmodernejšie techniky nedokážu zvládnuť analyzovať všetky typy ciest alebo zvládnuť analyzovať konkrétne prostredie a ich algoritmy, dokážu spracovať iba priaznivé podmienky a pri rôznych iných podmienkach nemusia fungovať. Algoritmus sa napríklad správa dobre v prostredí s diaľnicami (štruktúrované cesty) a vytvára neprijateľné prognózy snímok s cestami vo vidieckych oblastiach. Tiež niektoré z existujúcich algoritmov neboli natrénované, aby brali do úvahy zákruty, zasnežené, daždivé cesty a podmienky, ktoré úplne menia farby na scéne. Preto je tento problém stále otvorený a je potrebné ho vyriešiť v neštruktúrovaných prostrediach [3]. V rôznych prostrediach sa systém potrebuje zorientovať v priestore, zistiť trasu zvyčajne pomocou značení, ktoré sú zachytené kamerou vozidla. Následne prebieha interpretácia daných obrazov a extrakcia informácií. Z mnohých metód extrakcie bol veľký záujem o detekciu jazdného pruhu z cestných snímok.

Detekcia jazdného pruhu je základnou implementáciou vo vývoji inteligentných vozidiel, ktorá priamo ovplyvňuje správanie vozidla pri jazde. Na základe jazdného pruhu je možné určiť efektívny smer jazdy pre inteligentné vozidlo a poskytnúť presnú polohu vozidla. Tieto vlastnosti významne prispievajú k zlepšeniu účinnosti a bezpečnosti jazdy [1].

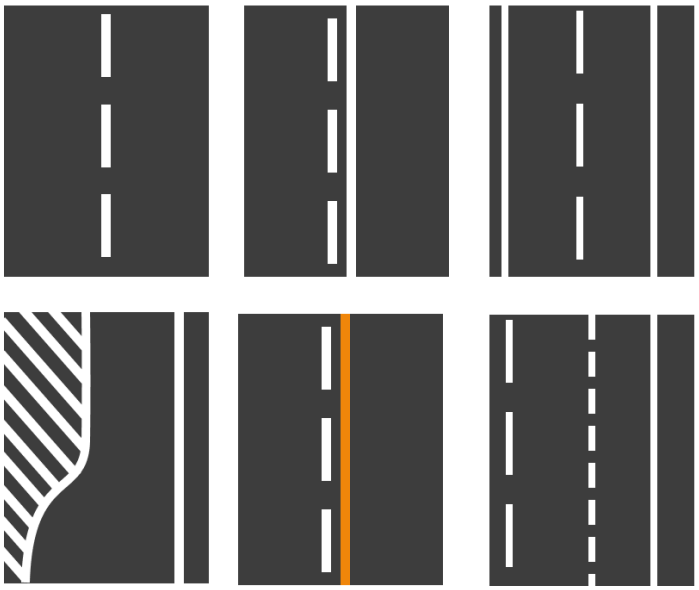
2.1 Faktory ovplyvňujúce analýzu jazdných pruhov

Aby bol vyvinutý kvalitný detektor, je potrebné ho odsimulovať v rôznych podmienkach, s rôznym typom ciest a cestných čiar, ktoré reálne tvoria cestnú infraštruktúru. Medzi faktory ovlyvňujúce analýzu jazdných pruhov patrí:

- **Dvojprúdová cesta s jednosmernou premávkou** - je lokalizovaná v mestských častiach s väčšou frekvenciou dopravy ako bývajú cesty prvej triedy, rýchlostné cesty a diaľnice. Pri rýchlostných cestách a diaľniciach bývajú cesty doplnené o odstavný pruh v prípade poruchy vozidla. Preto daný detektor môže naraziť na viac pruhov z pravej a ľavej strany.
- **Cesta so zákazom predbiehania** - v neprehľadných úsekoch alebo prudkých stúpaniach sa umiestňuje zákaz predbiehania na úseky, kde vodič predbiehajúceho vozidla nemá možnosť zaznamenať protiidúce vozidlo. Tieto úseky sa vyznačujú plnou čiarou od ľavého boku vozidla. Za touto čiarou môže nasledovať ďalšia plná čiara alebo prerušovaná čiara. Preto detektor musí byť schopný sa zamerať na signály čiary, ktorá mu je najbližšie a správne vyhodnotiť aktuálnu situáciu.
- **Odbočovací pruh** - diaľnice, rýchlostné cesty alebo cesty prvej triedy, ktoré disponujú MUK, znamenajú pre vozidlo a detektor zvýšený počet jazdných pruhov.

Ďalej treba brať do úvahy :

- **Cestu v rekonštrukcii** - stavebné úpravy na ceste sú zväčša vyznačované oranžovými alebo červenými pruhmi.
- **Vplyv počasia na cestu** - pri zlých poveternostných podmienkach sa môže na ceste objaviť súvislá vrstva vodnej plochy alebo snehovej prikrývky, ktorá má za následok zvýšený počet detekovaných plôch. Pri slnečných dňoch býva častokrát nastavená citlivosť kamery tak, aby potláčala svetlo, čo nie je vhodné pri náhodnom prechode tieňom, ktorý spôsobí dočasnú slepotu kamery a tým aj zvýšené riziko nebezpečia.
- **Neovplyvniteľnosť odleskami** - nočné zábery alebo slnečné svetlo môžu spôsobiť odlesk, ktorý zmätie detektor čiar, čo vedie k chybnej detekcii. Tento prípad môže nastať pri nočnej detekcii, kde svetlá prichádzajúceho oprotiidúceho vozidla v odlesku od čelného skla spôsobia vytvorenie novej čiary, ktorá bude detekovaná ako správna čiara, čo rozšíri jazdný pruh a tým pádom môže prejsť do prostismeru.
- **Prechádzanie medzi svetelnými podmienkami** - intenzita svetla v jeden okamih a jeho rýchla zmena sú časté situácie na ceste. Pri jazde cez tunel sú zväčša používané svetlá, ktorých tepelný efekt je silno sfarbený do oranžova. Pri výjazde z tunela sa celá svetelná scéna mení a svetlo má iné vlastnosti.



Obr. 2.1: Rôzne typy čiar na cestách

Obr. 2.1 zobrazuje výber čiar na ceste, s ktorými je možné sa stretnúť v premávke na európskych cestách. Tento obrázok slúži ako príklad, nezobrazuje všetky možné typy a situácie. Zachytáva obyčajnú cestu s jedným pruhom s možnosťou predchádzania, cestu s jednosmerným zákazom predbiehania, cestu s vyznačenou krajnicou, zúženie cesty, vyznačenie obchádzky a pripojovací pruh.

Kapitola 3

Detekcia jazdných pruhov

V súčasnosti je detekcia jazdných pruhov založená hlavne na vizuálnych senzoroch. Vizuálne senzory sa v podstate stali „očami“ inteligentného automobilu a prostredníctvom kamier snímajú scény pred vozidlami [1]. Sieť vizuálnych senzorov predstavuje sieť priestorovo rozmiestnených zariadení - inteligentných kamier, ktoré sú schopné spracovávať a spájať obrázky scény z rôznych hľadísk do podoby užitočnejšej ako jednotlivé obrázky. Sieť sa obvykle skladá zo samotných kamier, ktoré majú určité možnosti lokálneho spracovania obrazu, komunikácie a ukladania z jedného alebo viacerých centrálnych počítačov, kde sa obrazové údaje z viacerých kamier ďalej spracúvajú a spájajú [4] [5]. Zároveň vizuálne senzory majú široké spektrum rozsahu odozvy, dokážu vidieť infračervené lúče, ktoré sú neviditeľné voľným okom. Táto vlastnosť pozoruhodne zväčšuje priestor videnia ľudských bytostí. Dnes je možné rozdeliť riešenie na dve triedy prístupu k analýze tohoto problému a to je spracovanie obrazu pomocou metód bez učenia, alebo spracovanie obrazu pomocou neurónovej siete.

3.1 Metódy bez učenia

Tieto metódy využívajú na detekciu jazdného pruhu sériu filtrov tak, aby obraz po odfiltrovaní nepotrebných vecí z obrazu obsahoval len jazdné pruhy a tým bola umožnená bezchybná detekcia. Princípy preberané v tejto podkapitole využívajú množsto funkcií s OpenCV a taktiež majú veľa vecí spoločných.

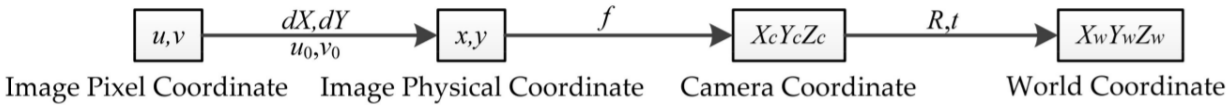
3.1.1 Algoritmus detekcie jazdného pruhu pre inteligenté vozidlá

Práca autora [1] je zameraná na predstavenie postupov na detekciu jazdného pruhu a to bez použitia neurónovej siete.

Kalibrácia kamery - pre potreby jasnej a kvalitnej snímky potrebuje kamerový systém sústrediť veľké množstvo svetla v momente vytvorenia snímky [1]. Za týmto účelom sa používa objektív, ktorý je mierne vypuklý a dokáže sústrediť naraz veľké množstvo svetla. To zabezpečí ohýbanie

lúčov svetla na senzor kamery. Efekt, ktorý je vytvorený objektívom je možné pozorovať na okrajoch snímky, kde je obraz skreslený. Reálna cesta je trojrozmerný priestor, avšak snímka, ktorú poskytne kamera je dvojrozmerná. Preto bolo potrebné vytvoriť geometrický model medzi svetlom a súradnicami a vykonať kalibráciu kamery prostredníctvom charakteristických parametrov, aby bolo skreslenie čo najviac eliminované.

Ak je obraz skreslený, bod P je projekčný na rovine obrazu podľa ideálneho lineárneho modelu a bod d je projekčný bod bodu P v rovine obrazu, kde dôjde k skresleniu obrazu. Dvojrozmerný obraz zachytený kamerou sa musí postupne prevádzať medzi príslušnými súradnicami, aby sa uskutočnila súradnicová transformácia určitého pixelového bodu v obraze zo súradnice pixelu na svetovú súradnicu. Tento postup je vidieť na obr. 3.1



Obr. 3.1: Zobrazenie prevodu

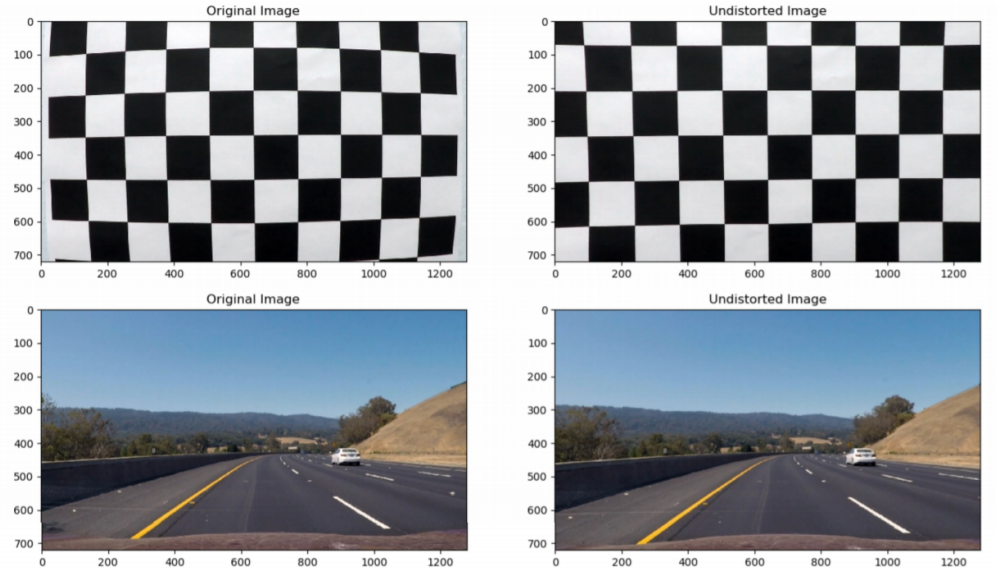
Odstránenie skreslenia obrazu - skreslenie obrazu sa všeobecne delí na radiálne a tangenciálne skreslenie[1]. V porovnaní so skutočnou scénou na ceste sa bežné skreslenie okrajov jazdných pruhov vozidiel a pozadia nazýva radiálne skreslenie. Matematický model radiálneho skreslenia 3.1 je definovaný ako :

$$\begin{cases} u' = u(1 + 5_1 r^2 + k_2 r^4 + k_3 r^6) \\ v' = v(1 + 5_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \tag{3.1}$$

kde r^2 je súradnica bodu d v pixeloch

$$r^2 = x^2 + y^2, (u', v') \tag{3.2}$$

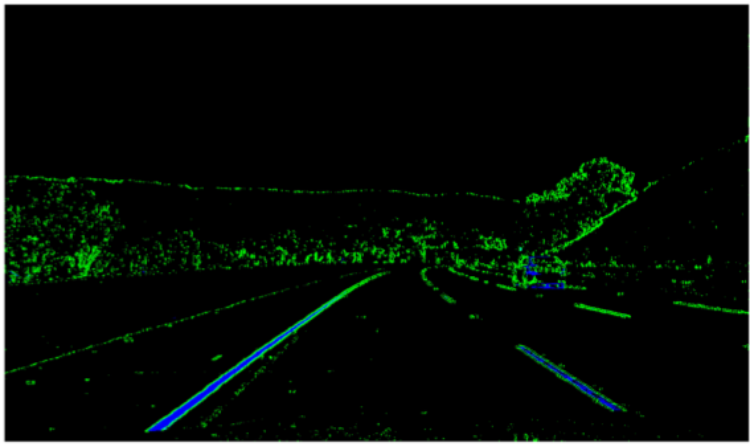
Podľa rovnice 3.2 a 3.1 je zrejme, že skreslenie obrazu súvisí s piatimi parametrami - k1, k2, k3, p1 a p2 - ktoré sa súhrnne nazývajú koeficienty skreslenia. Kalibráciou kamery a získaním koeficientu skreslenia je možné skreslený obraz korigovať. Ako príklad možno uviesť štúdiu podľa [1], pri ktorej sa použila šachová kalibračná doska na odstránenie skreslenia.



Obr. 3.2: Korekcia skreslenia obrazu [1]

Obr. 3.2 zobrazuje korekciu skresleného obrazu, pri ktorej ľavá a pravá polovica sú pôvodné obrázky a neskreslené obrázky. Šachovnica bola zložená z čiernej a bielej mriežky s pravidelnými tvarmi, aby bolo možné jednoducho nájsť súradnice rohových bodov a jednoducho sledovať korekciu obrazu. Po určení súradníc rohov bola vytvorená transformačná matica na mapovanie skreslených bodov na neskreslené body a skreslenie obrazu bolo odstránené pomocou transformácie súradníc.

Detekcia hrany / Sobel detektor - informácie o okraji sú dôležitou vlastnosťou detekčného obrazu, preto je potrebné zvoliť vhodný detekčný algoritmus, ktorým sa odstráni veľké množstvo zbytočných údajov a je prospešný na určenie základného obrysu jazdného pruhu [1]. Medzi bežne používané algoritmy detekcie okrajov patria globálne extrakčné metódy založené na kritériu minimalizácie energie (napríklad fuzzy teória a neurónové siete) a derivačné metódy hrany, ktoré používajú diferenciálne operátory (napríklad operátory Canny, Prewitt, LOG a Robert a iné). Tradičný algoritmus detekcie jednej hrany má však veľa problémov vrátane príliš širokého rozsahu detekcie, slabej protihlukovej interferencie a predĺženého času výpočtu. Preto bol autorom [1] zvolený algoritmus, ktorý superponuje prahovú hodnotu operátora Sobel a farebný priestor HSL. HSL je farebný priestor, ktorý je možné previesť z bežnej farby RGB priestoru. Jednotlivé časti H, S a L označujú odtieň, sýtosť a svetlosť. Operátor Sobel je typom diskrétneho operátora rozdielu prvého rádu. Vplyv susedných bodov pre aktuálny bod pixelu sa líši. Porovnávanie na stupnici šedej operácie boli vykonávané na susedných pixeloch za účelom získania gradientu a normálnych vektorov pixelových bodov, ktoré možno použiť na výpočet približnej hodnoty funkcie jasu obrazu. Operátor Sobel vykonáva detekciu hrán na detekovanom obrázku z horizontálnej a vertikálnej polohy v smeroch a dokáže dostatočne filtrovať interferenčný šum s vylepšeným efektom spracovania obrazu. Obr. 3.3 zobrazuje scénu po prevedení funkcie sobel. Scéna je vo farebnom spektre HSL.



Obr. 3.3: Obraz po prevedení funkcie Sobel vo farbach HSL [1]

Určenie regiónu záujmu - v článku [1] je nevyhnutný krok pre správny chod fungovania algoritmu či sa jedná o zníženie výpočtovej zložitosti, kedy nemusí detektor prechádzať každý pixel, ale len istú časť, ale taktiež aj pre zníženie chybovosti a elimináciu nevhodných objektov z obrazu, kedy sa zvolený detektor zameria len na presne vymedzenú časť. Napríklad ak pri obrázku 240x320 výsledný obrázok, ktorý bude ďalej spracovávaný po uplatnení oblasti záujmu bude pozostávať z rozlíšenia 120x160. Avšak pri detekcii čiar na ceste býva obvykle, že daný objekt záujmu býva často trojuholník alebo lichobežník v závislosti od polohy kamery. Nastavenia tohoto objektu záujmu sú individuálne ako je vidieť na obr. 3.4

Obr. 3.4: Región záujmu zobrazený na snímke skutočnej cesty

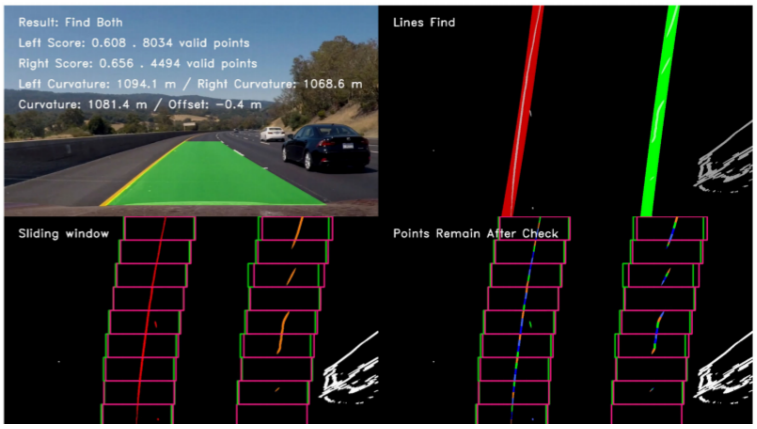
Transformácia inverznej perspektívy - autor publikácie [6] vysvetľuje prevod medzi 3D scenou a 2D obrazom. Čím bližšie ste k fotoaparátu, tým objekt vyzerá väčší a naopak. Paralelné čiary pruhu sa spájajú do bodu vo vzdialenom zornom poli nazývaného úběžník. Vzdialenosť medzi susednými jazdnými pruhmi v blízkosti úbežníka sa postupne zmenšuje, čo škodí účinnej detekcii jazdného pruhu. Transformácia inverznej perspektívy je založená na inverznej transformácií súradníc

zo svetovej súradnice na súradnicu obrazu, ktorá transformuje perspektívny obraz do leteckého pohľadu a obnoví paralelný vzťah medzi jazdným pruhom [6]. Obr. 3.5 predstavuje letecký pohľad jazdného pruhu získaného transformáciou inverznej perspektívy.

Obr. 3.5: Inverzná transformácia perspetívy [1]

Pri dynamickej jazde však musí byť použitá efektívna detekcia jazdného pruhu v prostredí, pri zachovaní rýchlosti rozpoznávania s dostatočne vysokou presnosťou. Autor najskôr vykonal aplikáciu masky. Potom na základe B-spline tretieho rádu model krivky. Bol použitý algoritmus RANSAC na dosiahnutie presného prispôsobenia a výpočtu zakrivenia jazdného pruhu.

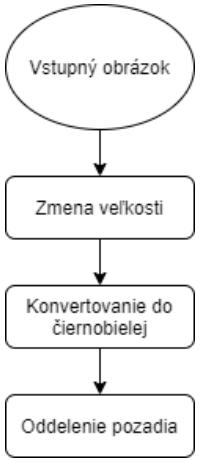
Autorovi [1] sa podarilo dosiahnuť 98.49 percentnú, úspešnosť pri testovaní používal dataset, ktorý obsahoval 4768 snímkov. Úspešnosť jeho detekcie a výstup programu je možné vidieť na obr. 3.6.



Obr. 3.6: Výstup programu [1]

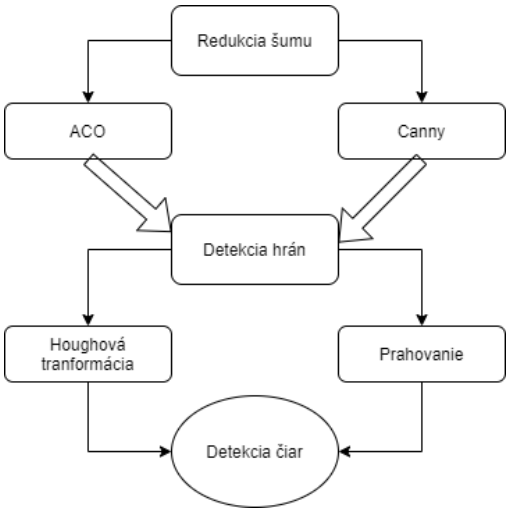
3.1.2 Detekcia jazdných pruhov pomocou Cannyho detektora a optimalizácie kolónií mravcov ACO

Pôvodný vedecký článok "Road Lane Detection with Improved Canny Edges Using Ant Colony Optimization,"bol publikovaný autormi P. M. Daigavane and P. R. Bajaj v roku 2010 [7] a zachytáva hybridný postup využívajúci Ant Colony optimalizáciu (ACO) a Cannyho detekciu okrajov a niekoľko ďalších postupov s cieľom detekovať jazdné pruhy na diaľnici. Všeobecne sa uznáva, že detektor Canny je pravdepodobne jedným z najlepších algoritmov detekcie okrajov, pokiaľ produkuje hrany, ktoré sú široké jeden pixel, avšak trpí nesúrodosťou detekovaných hrán v miestach, kde sa spájajú okraje. Vstupnými údajmi bol farebný obrázok zhotovený z farebného fotoaparátu a uložený do pamäte počítača. Detekčný systém načítal obrázok z pamäte a spracoval ho. Za účelom získania dobrých odhadov jazdných pruhov a zlepšenia rýchlosti algoritmu sa pôvodná veľkosť obrázka zmenšila na 255 x 255 pixelov. Ďalším krokom bolo prevedenie obrázka do odtieňa sivej, čo minimalizovalo niektoré rušivé farebné vplyvy. Táto funkcia transformovala v tomto prípade 24-bitový, trojkanálový farebný obraz na 8 bitový. Toto predspracovanie obrázka je možné vidieť na obr. 3.7



Obr. 3.7: Diagram predspracovania obrázka

Na odstránenie šumu sa použil stredný filter, ktorý eliminoval šum, ale zachoval hrany. Po aplikovaní šumového filtra sú na obrázok aplikovaé dve funkcie, štandardná Canny a ACO. Tento postup je možné vidieť na obr. 3.8



Obr. 3.8: Diagram algoritmu s ACO

Canny detektor - algoritmus Canny možno popísať viacerými krokmi.V publikácii [8] ho autor popisuje. Algoritmus zvyčajne používa na vyhladenie dvojrozmernú Gaussovu funkciu 3.3, ktorá je popísaná nižšie.

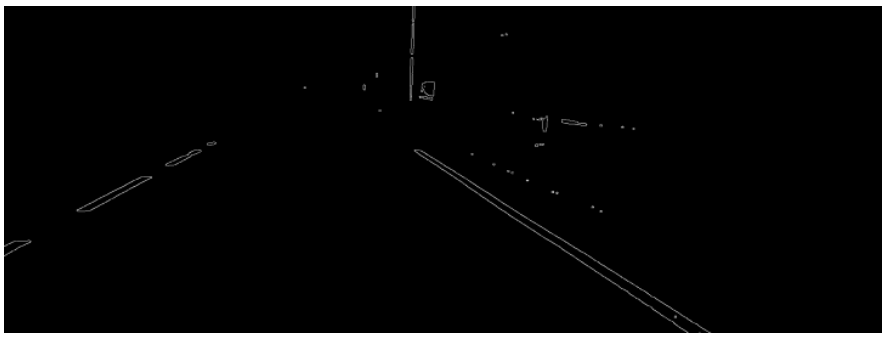
$$G(x, y) = exp[-(x^2 + y^2)/2\sigma^2]/2\pi\sigma^2 \tag{3.3}$$

Ďalším krokom je výpočet veľkosti a smer gradientu obrazu. Získava amplitúdový gradient a smer pomocou diferenciálnych operátorov. Algoritmus zoberie obmedzenú 2x2 susednú oblasť a vypočíta smer gradientu obrazu. Po získaní obrázka s gradientovou veľkosťou je potrebné pomocou techniky non-maximum suppression zistiť hrany v obrázku. Proces NMS zaručí, že každá hrana bude mať šírku jedného pixela. Cannyho algoritmus využíva maticu 3x3 susednej oblasti, ktorá sa skladá z ôsmich smerov. Následne vykonáva interpoláciu s veľkosťou gradientu pozdĺž smeru gradientu. Ak je vypočítaná veľkosť pre pixel M [i, j] väčšia ako dva interpolačné výsledky v smere gradientu, je bod pridaný do zoznamu hrán, inak bude označený ako bod nepatriaci do hrany.

Algoritmus Canny si privlastňuje metódu dvojtého prahovania pre výber bodov hrán po prevedení NMS. Pixel, ktorého veľkosť gradientu je vyššia ako prahová veľkosť rozsahu je určený ako bod hrany a naopak pixel, ktorého veľkosť gradientu je nižšia ako prahová veľkosť rozsahu je označený ako bod, ktorý nepatrí hrane. Ostatné body sa označia ako kandidáti na hrany a ak sa spoja s už existujúcimi bodmi hrany patria do danej hrany. Aj keď sa tradičný algoritmus stále prakticky používa, existujú dva aspekty na jeho zlepšovanie. Prvý aspekt je pevne určená veľkosť matice 2x2 pre výpočet gradientu obrazu. Tento spôsob je jednoduchší na výpočet, ale citlivejší na šum. Druhým aspektom je, že dvojité prahovanie je pri Cannyho algoritme nastavené pevnou hodnotou. Pri obrázkoch bohatých na informácie môže dôjsť k strate hrany.

Cannyho algoritmus bol široko používaný na detekciu okrajov obrázu. Stále však existujú nedos-

tatky, ktorými sa vyznačuje. Jedným z nedostatkov je to, že Gaussovo filtrovanie je citlivé na šum a to vedie jednoducho k vytvoreniu izolovaných okrajových bodov a teda k vytváraniu tzv. pseudo okrajových bodov a falošných hrán. Ďalšou nevýhodou je, že dvojité prahovanie má fixne nastavené hodnoty prahov a tým je daný algoritmus neadaptabilný [9] [8]. Výstup algoritmu je vidieť na obr. 3.9, kde bola táto funkcia použitá na snímku lokálnej cesty.



Obr. 3.9: Vykreslenie cesty po aplikácií Cannyho algoritmu

Prahovanie - je druh procesu, kedy hodnota pixelu sa zmení na jedna alebo 0 podľa určeného prahu [10]. Na obrázok, ktorý je prevedený do čiernobielej je aplikovaná táto funkcia. Thresholding je transformácia zo vstupného obrázka f(i,j) na výstupný obrázok g(i,j), kde T je hodnota prahu. Táto funkcia je popísaná na vzorci 3.4.

$$g(i,j) = \begin{cases} 1 & pref(i,j) \geq T \\ 0 & pref(i,j) < T \end{cases} \tag{3.4}$$

ACO (Ant colony optimization) - vstup do ACO predstavoval výstup z detektora Canny, čo je binarizovaný obraz tenkých hrán, z ktorých každá hrana je široká 1 pixel. ACO je prírodou inšpirovaný optimalizačný algoritmus inšpirovaný prírodným javom, charakterizujúcim mravce ukladajúce feromóny na zem, čo umožňuje vyznačiť cestu, ktorá by mala byť nasledovaná ďalšími členmi kolónie. Hlavným cieľom zavedenia ACO algoritmu bolo zvládnutie detekčného problému súvisiaceho s neúplnou detekciou všetkých okrajových informácií pomocou Cannyho detektora. Navrhovaný prístup využíval niekoľko mravcov, ktoré sa pohybovali po obraze riadenom miestnou variáciou hodnoty intenzity obrazu a vytvorili feromónovú matricu, ktorá predstavovala informáciu o okraji na každom pixelovom mieste. Každým mravcom počas iterácie sa neustále pohybovalo pomocou pravdepodobnostného prístupu, kým sa nedosiahol ďalší úsečkový segment.Extrakcia pruhov prebehla pomocou Houghovej transformácie.

Houghová tranformácia - (angl. Hough transform) je segmentačnou technikou spracovania obrazu, ktorá sa používa, keď je potrebné detekovať objekty so známym tvarom hranice [11]. Bola navrhnutá Paulom Houghom v roku 1962, neskôr bola upravovaná a rozšírená. V roku 1972 Richard Duda a Peter Hart dali sieti pomenovanie všeobecná Houghova transformácia na základe detekcie

všeobecných parametrických kriviek v obraze. HT slúži na hľadanie takých objektov, ktoré možno jednoducho popísať analytickou geometriou ako sú priamky, prípadne kružnice a elipsy. HT využíva na minimalizáciu počtu bodov prehľadávaných v obrazoch detektory – hranové filtre ako Cannyho, Sobelov alebo Prewitovej filter. Ďalším dôležitým krokom je vytvorenie všetkých možných tvarov popísateľných parametrami. Pôvodný návrh využíval súradnicové parametre, avšak neumožňoval správne vyhľadať zvislé priamky, čo viedlo k použitiu polárnych priamok. Rovnicový tvar smernice priamky je zobrazený 3.5

$$y = kx + q \tag{3.5}$$

k - predstavuje smernicu priamky. q - predstavuje priesečník s osou y.

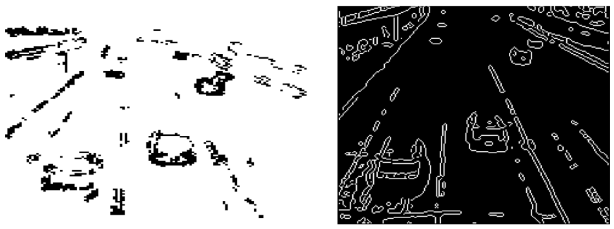
$$\begin{aligned} y &= -\frac{\cos\theta}{\sin\theta}x + \frac{r}{\sin\theta} \\ r &= x\cos\theta + y\sin\theta \end{aligned} \tag{3.6}$$

Podľa akumulátorového priestoru Hough boli vrcholy označené a predstavovali v reáli existenciu priamok. Horizontálna os - predstavuje polohu čiary v obrázku relatívne a vertikálna os - predstavujúca vzdialenosť čiar zodpovedajúca rohu roviny osi obrazu. Navyše kvôli chybám v nedokonalosti v detekcii hrán sa zvyčajne vyskytnú chyby v priestore akumulátora, čo môže viesť k tomu, že nebude potrebné hľadať vhodné vrcholy a teda príslušné riadky.

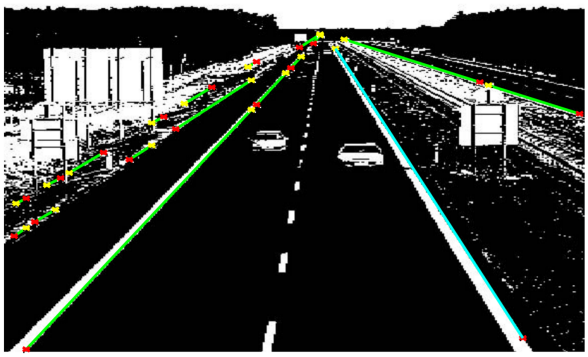
Po vytvorení všetkých možných tvarov a detekcii ich hrán sa vyšetruje prítomnosť bodov obsiahnutých v tvaroch tzv. proces voting procedure. Každý obsiahnutý bod získa hlas. Hlasy sú ukladané do akumulátora - akumulačnej matice. Akumulátor je charakterizovaný ako pomocný diskrétny priestor parametrov (q, k) reprezentovaný maticou, možno vidieť na matici 3.7.

$$\begin{bmatrix} k_{min}, q_{min} & \cdots & k_{min}, q^{max} \\ \vdots & \vdots & \vdots \\ k^{max}, q_{min} & \dots & k^{max}, q^{max} \end{bmatrix} \tag{3.7}$$

Posledným krokom je výber maxím akumulátora, pričom počet vybratých maxím je zvolený používateľom, na základe maxím sa vykreslia tvary [12] [13] [14]. Experimentálne výsledky ukazujú, že navrhovaná schéma bola robustná ako zobrazuje obr. 3.10, avšak prístup potrebuje ešte viac výskumu [7]. Samotné porovnanie Canny a ACO je možné vidieť na obr. 3.10 a výsledný obraz je zobrazený na obr. 3.11.



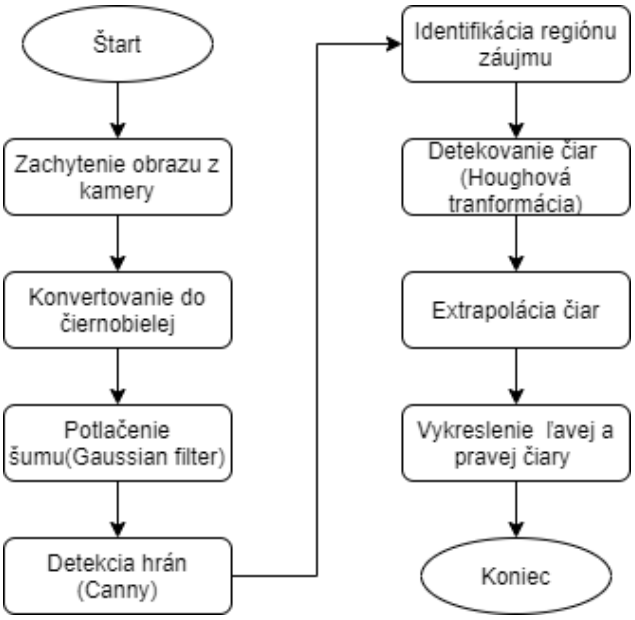
Obr. 3.10: Porovnanie Canny a kolónie mravcov[7]



Obr. 3.11: detekcia čiar daným algoritmom [7]

3.1.3 Zjednodušený detektor jazdného pruhu - LaneRTD

Tento príspevok [15] je zameraný na riešenie problému detekcie jazdných pruhov a to pri nižšom výpočtovom výkone a real time detekcii. Tento spôsob môže byť ďalším krokom k vyhliadkam na autonómne riadenie vozidla, hlavná inovácia navrhovaného riešenia je jemná rovnováha medzi rýchlosťou a nízkymi požiadavkami na zdroje počítača tak, aby tým nebola ohrozená spoľahlivosť a odolnosť voči chybe. Jednou z výhod algoritmu LaneRTD je, že používa iba jednu CCD kameru. Táto kamera by mala byť pripevnená na zrkadle predného okna vozidla, aby zachytila čelný pohľad na cestu [16].



Obr. 3.12: Algoritmus detekcie

Algoritmus je ilustrovaný vývojovým diagramom na obr. 3.12 a implementovaný pomocou jazyka Python a knižnice OpenCV. Postup pozostáva z čítania testovacích obrázkov v abecednom poradí, konverzie farebného testovacieho obrázka na sivú pomocou funkcie OpenCV „ctvColor“, filtrovania šumu, ktoré sa vykonáva pomocou funkcie OpenCV „GaussianBlur“. Detekcia a extrakcia okrajov sa vykonáva pomocou funkcie OpenCV „Canny“, ktorá je známa ako Cannyho algoritmus. Identifikácia záujmového regiónu je implementovaná maskovaním lichobežníkovej oblasti v obraze s detekovanými okrajmi na vytvorenie obrazu a má iba čiary zodpovedajúce cestným pruhom. Priame čiary sú identifikované pomocou Houghovho algoritmu a polárnych súradniciach pomocou funkcie OpenCV HoughLinesP. Po úspešnej detekcii jazdného pruhu na stacionárnom obrázku, ktorý je popísaný v predchádzajúcich krokoch je pri sledovaní jazdého pruhu v reálnom čase potrebné aplikovať tieto kroky na každý snímok, ktorý príde z kamery. Draw_Lines() funkcia kreslenia čiary je implementovaná na pripojenie traťových segmentov pre každú jazdnú dráhu (vľavo alebo vpravo), aby sa vytvorila jedna plná čiara, ktorá sleduje skutočnú hodnotu čiary pruhu na obrázkoch. Úsečky sú produkované Houghovou transformáciou. Klasifikácia sa robí na základe sklonu úsečky. Ak sklon je kladný a leží medzi 0,4 => 1,0 a potom segment patrí do triedy ľavého riadku, a ak je záporný a -0,4 => -1,0 potom patrí do triedy pravého riadku. Všetky klasifikované segmenty ľavej a pravej čiary sú vypočítané a uložené spolu s ich sklonmi. Pre každú triedu (ľavú a pravú) sa používa technika zarovnávania priamok, pričom sklon každej úsečky sa berie ako indikácia dobrého alebo zlého (šumového) segmentu [15]. Výsledný obraz je zobrazený na obr. 3.13.

Obr. 3.13: Zobrazenie na reálnej ceste [15]

Vyvinutý algoritmus LaneRTD sa ďalej testoval na mnohých obrázkoch predstavujúcich rôzne scenáre. Prezentované výsledky ukazujú, že algoritmus funguje veľmi dobre za rôznych podmienok. Na testovanie robustnosti a validácie vyvinutého algoritmu sa aplikovalo niekoľko vzoriek videa v reálnom čase a použili sa rôzne jazdné podmienky. LaneRTD sa ukázal byť veľmi robustný až na jednu podmienku rozptýlených oblasti tieňov, ktoré zavádzajú algoritmus tak, aby produkoval nepresné detekcie čiary jazdného pruhu. Algoritmus sa ukázal byť aj prijateľne rýchlym pri realizácii v reálnom čase. Použil sa procesor Intel Core i5 s 1,6 GHz a 8 GB RAM. Najnižšia nameraná rýchlosť spracovania je 11 snímok za sekundu, čo je postačujúce pre presnú detekciu jazdných pruhov avšak v porovnaní s novšími prístupmi s využitím konvolučných neurónových sieti nie je použiteľná v súčasných systémoch ADAS. Ďalším nedostatkom je, že algoritmus detekuje iba priame jazdné pruhy. Na zvládnutie zákrut (alebo zakrivenie jazdných pruhov) je stále potrebné viac skúmaní, taktiež zvládnutie cesty do kopca alebo z kopca.

3.2 Metódy s učením

Detekcia pomocou neurónovej siete je dnes veľmi využívaná, kde sa daný detektor sám učí. Metodiky založené na hlbokom učení môžu účinne zvýšiť presnosť a odolnosť detekcie jazdných pruhov. Algoritmy však majú vyššie hardvérové požiadavky a štruktúry výcvikových modelov, sú príliš zložité, preto stále existujú určité obmedzenia a je potrebné ďalšie zdokonalenie algoritmu detekcie jazdných pruhov. Prístupy založené na neurónových sieťach a hlbokom učení a najmä konvolučné neurónové siete (CNN) stimulujú sľubný smer výskumu aj napriek ohromnej výpočtovej réžii [1].

3.2.1 Konvolučné neurónové siete

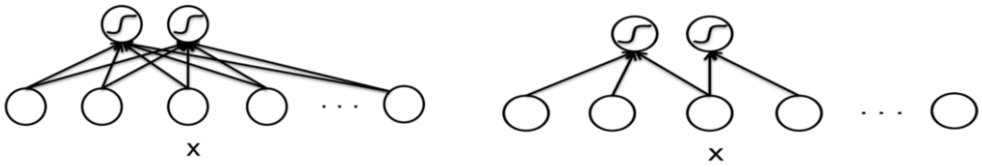
Neurónové siete našli uplatnenie pri rôznych riešeniach problémov so spracovaním signálu [17]. Skladajú sa zo základných jednotiek, ktoré môžno prirovnať k ľudským biologickým neurónom. Tieto jednotky sú navzájom prepojené spojeniami, ktorých sila sa môže meniť v dôsledku procesu učenia

alebo algoritmov. Každá z týchto jednotiek integruje nezávisle informácie poskytované svojimi synapsami a tým vyhodnocuje stav svojej aktivácie. Jednotková odozva je lineárnou alebo nelineárnou funkciou jej aktivácie [18]. Teória neurónových sietí čerpá z neurofyziologických znalostí. Pokúša sa vysvetliť správanie systému, ktorý funguje na princípe spracovania informácií v nervových bunkách, preto možno označiť neurónové siete ako modely mozgu bez mysle„brain without mind “.

Neurón alebo nervová bunka je najvýznamnejší stavebný a funkčný prvok nervového tkaniva. Hlavnou funkciou nervového systému je riadiť organizmus. V nervovom systéme sa informácie prenášajú na základe zmien membránového potenciálu neurónov [19]. V porovnaní s ľudským neurónom sa pomocou počítačovej technológie darí nasimulovať omnoho rýchlejší neurón. Problémom je však počet neurónov a množstvo ich spojení v mozgu, ktorý v konečnom dôsledku určuje silu celej neurónovej siete[20].

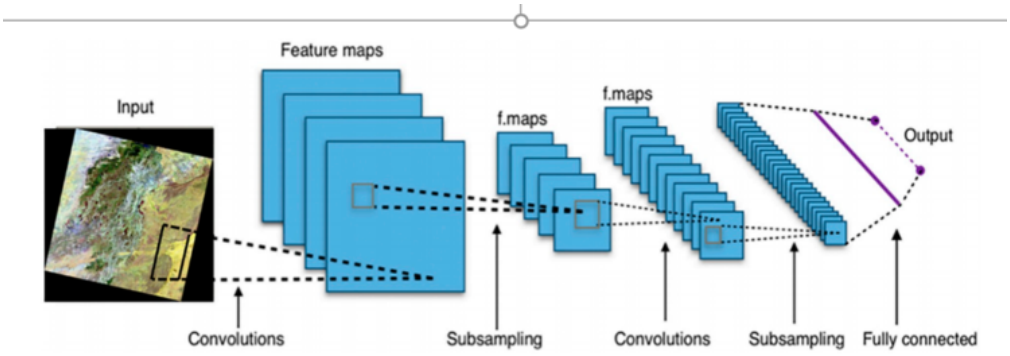
Konvolučné neurónové siete angl. Convolutional Neural Network (CNN) sa zaraďujú medzi hlboké viacvrstvové neurónové siete. Na rozdiel od bežných neurónových sietí obsahujú konvolučnú vrstvu. CNN sa vyznačujú rovnakými parametrami ako všeobecné neurónové sústavy. Sú charakterizované prostredníctvom váh a hodnoty váh bývajú predmetom učenia. Predstavujú veľmi efektívny nástroj počítačového videnia v oblastiach rozpoznávania obrazu a klasifikácie [21]. Postavené sú na základoch neocognitronovej siete, ktorú navrhol K.Fukushima. Neocognitronové siete sú charakteristické hierarchickou viacvrstvovou štrukúrou, ktorú možno prirovnať k štruktúre vizuálného cortexu - mozgovej kôry, ktorá je zodpovedná za zrakové vnemy. Teoreticky ju tvoria jednoduché bunky, nasledujú komplexné bunky, nízko úrovňové hyperkomplexné bunky a vysoko úrovňové hyperkomplexné bunky. Spojenia medzi jednoduchými a komplexnými bunkami sú porovnateľné ako medzi nízko a vysoko úrovňovými bunkami [22]. Tento typ siete dokáže na základe podobnosti vzorov identifikovať objekty a to bez ohľadu na posunutie pozícií, zmenu veľkosti alebo deformáciu [23].

Vznik CNN sa datuje do roku 1980, ale do popredia sa dostali až v roku 2012. V tomto roku sa metóda pod názvom Alexnet založená na konvolučnej sieti zúčastnila súťaže Imagenet Visual Recognition Challenge, kde súťažili algortimy pre klasifikáciu obrazu a súťaž s vysokým náskokom vyhrala. Dnes sa CNN využívajú v rozličných odvetviach. Najčastejšie sú využité v systémoch na rozpoznávanie tváre, dopravných značiek a ďalších objektov v robotike či v automobilovom priemysle [21]. Medzi typické vlastnosti konvolučných neurónových sietí možno zaradiť technológiu zdieľaných váh, schopnosť získať príznaky z recepčných polí a technológiu priestorového vzorkovania. CNN sú založené na princípe lokálnej konektivity, ktorá je využitá v celej sieti. V praxi to znamená, že každý neurón je napojený na malú časť zo vstupu. Tým sa výrazne znižuje nárast parametrov. Vo všeobecných neurónových sieťach sa každý neurón v prvej skrytej vrstve pripája ku všetkým neurónom vo vstupoch. Toto je zobrazené na obr. 3.14 [24].



Obr. 3.14: Všeobecná neurónová sieť vs. konvolučná neurónová siet

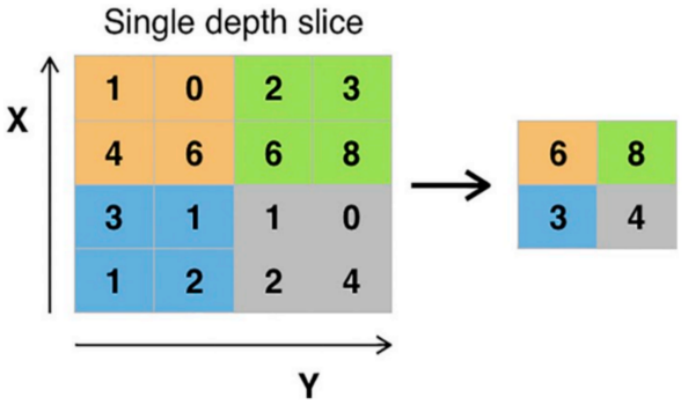
Konvolúcia - neuróny v konvolučnej vrstve počítajú základnú funkciu, nazývanú konvolúcia. Jadrom konvolúcie sú parametre (váhy). Jadro býva pomenované ako kernel alebo filter. Jedna konvolučná vrstva môže obsahovať niekoľko filtrov. Na základe počtu filtrov sa realizuje počet konvolúcií pri prechode danou vrstvou. Výstupný pixel konvolúcie je daný ako súčet hodnôt konvolučného jadra vynásobený hodnotami vstupného obrazu. Pri konvolúcii sa filter prikladá na vstup a navzájom sa násobia hodnoty prekrytia filtra a vstupu. Do príznakovej mapy sa postupne zaznamenávajú výsledky singulárných konvolúcií. Na záver konvolúcie obsahuje príznaková mapa na vstupe výsledky skalárneho súčinu celej plochy vstupu [25] [26]. Na zmenšenie veľkosti vstupu slúži subsamplingová vrstva. Nachádza sa za konvolučnou vrstvou. V subsamplingovej vrstve je možné aplikovať funkcie, ktoré slúžia na zmenšenie príznakových máp, ktoré predstavujú výstup konvolučnej vrstvy [27]. Medzi tieto funkcie môžeme zaradiť max-pooling, avg-pooling alebo lienárnu kombináciu neurónov. Hodnoty výstupov neurónov sú určené prostredníctvom aktivačnej funkcie z ich vnútorných potenciálov. Aktivačné funkcie patria medzi nelineárne funkcie a možno k nim zaradiť sigmoidnú funkciu, funkciu Tanh, ReLU, Leaky ReLU a Maxout [28]. Základnú architektúru konvolučnej neurónovej siete je možné vidieť na obr. 3.15.



Obr. 3.15: Základná architetkúra konvolučnej neurónovej siete [29]

Pooling vrstva - hlavnou úlohou pooling vrstvy v CNN je tzv. down-sampling teda redukovať zložitosť nasledujúcich vrstiev alebo tzv. podvzorkovanie pôvodného vstupu, výsledkom čoho je zmenšenie rozmerov vstupu. Pooling vrstva funguje na podobnom princípe ako konvolučná vrstva. Rovnako ako v konvulúčnej vrstve sú špecifickými parametrami veľkosť filtra a stride (krok). Na rozdiel od konvolučnej vrstvy sa veľkosť kroku volí tým spôsobom, aby nedošlo k prekrytiu oblasti

prikladania filtra. Celý proces možno chápať ako masku postupujúcu celým obrazom. Maska z danej matice bodov vyberie výslednú výstupnú hodnotu. Výstupná hodnota závisí od typu filtrov. Ak filter vyberie maximálnu hodnotu z danej oblasti vstupu ide o operáciu maxpooling. Najčastejšie využívanou metódou zmenšenia vstupu je max pooling. Ak filter priemeruje hodnoty, ide o operáciu AVG Pooling [28] [30]. Príklad je vidieť na obr. 3.16.

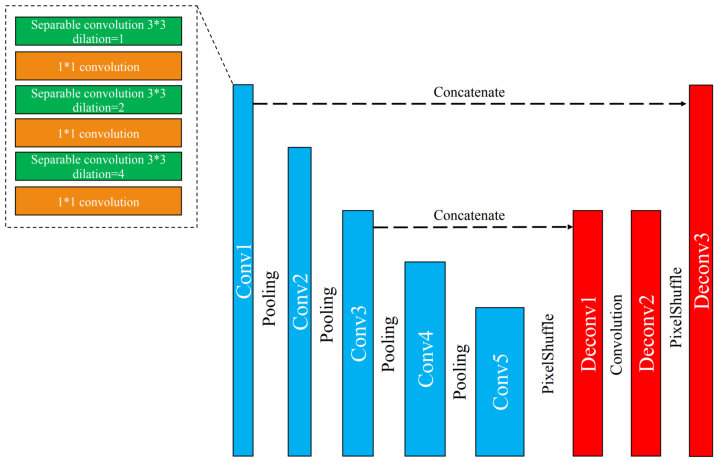


Obr. 3.16: pooling proces [29]

3.2.2 LaneNet: Detekcia jazdných pruhov v reálnom čase pre autonómnu jazdu

V článku [31] je navrhnutá hlboká neurónová sieť využívajúca metódu LaneNet, ktorá rozdeľuje detekciu jazdných pruhov na dve etapy: prvotná predikcia jedného pruhu a nasledná presná lokalizávia jazdneho pruhu z predchádzajúcej predikcie. Cieľ LaneNetu je postavený tak, aby detekoval iba čiaru jazdného pruhu, to však môže spôsobiť ťažkosti a nesprávnu detekciu, napríklad namiesto určeného jazdného pruhu môžu byť detekované rôzne označenia na ceste ako je šípka, odbočovací pruh a podobne . Napriek všetkým ťažkostiam sa ukázalo, že detekcia jazdných pruhov je robustná pri detekcii diaľníc aj mestských komunikácií. Vysoká rýchlosť a nízke výpočtové náklady ukazujú na perspektívne využitie v reálných cestných scénach.

Konkrétne sa pre návrh jazdných pruhov používa architektúra enkodér-dekóder, kde sa na rýchle kódovanie funkcií používajú vrstvené konvolučné vrstvy. Získaná návrhová mapa sa potom transformuje na súradnice okraja jazdného pruhu a privádza sa do druhého kroku, kde je umiestnená vysokorýchlostná sieť na lokalizáciu jazdných pruhov pozostávajúca z kodéra bodových prvkov a dekodéra LSTM umožňujúca robustnú lokalizáciu jazdných pruhov podľa rôznych scenárov. Na obr. 3.17 je vidieť architektúru siete na prvotnú predikciu jazdných pruhov .



Obr. 3.17: Sieť detekcie čiar [31]

Trénovanie siete pre navrhovane okraja jazdných pruhov je priame. Model na vstupe sníma obraz čelného pohľadu na vozidle a na výstup poskytuje mapu pravdepodobnosti okraja jazdného pruhu rovnakej veľkosti ako vstupný obrázok. Všetky parametre sú trénované s úplným dohľadom. Pre každý cvičný obrázok je poskytnutá anotačná mapa, kde „1“ označuje, že tento pixel predstavuje kladný bod okraja jedného segmentu jazdného pruhu. Celá sieť bola end-to-end optimalizovaná stochastickým gradientom zostupu. V každej anotačnej mape je uvedený počet kladne detekovaných bodov, ktorý je vždy oveľa menší ako počet neidentifikovaných bodov. V tréningovom procese sa dynamicky váži strata pozitívnych a negatívnych bodov podľa ich relatívnych množstiev. Po natrenovaní sieť vytvára priamo navrhovanú mapu okraja jazdného pruhu vstupného obrázka, kde hodnota každého pixelu naznačuje, že spoľahlivosť tohto pixela leží na okraji jedného segmentu jazdného pruhu.

Každá čiara jazdného pruhu je znázornená ako kvadratická funkcia, teda tvar každého jazdného pruhu je anotovaný tromi spojitými číslami. To však spôsobuje ťažkosti, v praxi trénovanie siete na predpovedanie parametrov každej kvadratickej funkcie funguje zle, pretože každý parameter je zvyčajne významne odlišného rádu. Sieť je potrebné trénovať tak, aby bolo možné predpovedať polohu bodov, kde sa križuje jazdná dráha s hornou, strednou a spodnou čiarou obrazu. V tomto prípade má vstupný obrázok veľkosť h × w, takže hodnoty, ktoré sieť predpovedá, sú tri hodnoty koordinácie X bodov, ktoré ležia na križovatkách jazdného pruhu s tromi vodorovnými čiarami Y = 0, Y = h / 2 a Y = h. Tieto tri hodnoty sa prenesú na kľúčové hodnoty v jazdnom pruhu. Prenesením tréningového cieľa z parametrov kvadratickej funkcie do kľúčových hodnôt sú predikované hodnoty relatívne podobných rádov, takže sa tréning stane stabilný a je zabránené konvergancii na zlom miestnom minime.

Obr. 3.18: Výsledok LaneNet [31]

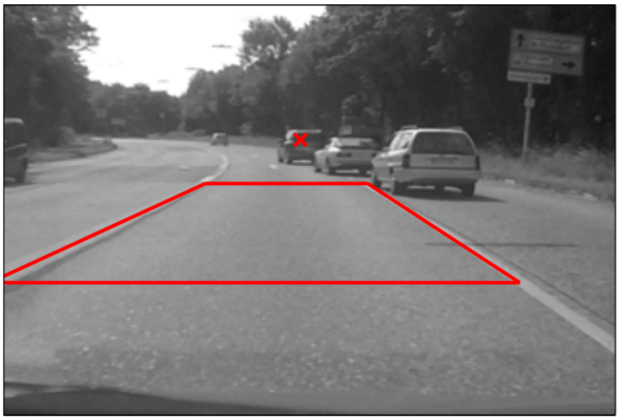
Na obr. 3.18 je možné vidieť výstup programu, kde je naľavo zobrazený obrázok bez detekcie. Obrázok v strede je s vykreslenými všetkými čiarami, krok po prvej predikcii. Obrázok napravo je farebné odlíšenie jednotlivých čiar a je to výstup programu po prevedení druhého kroku. Tento program bol testovaný na datasete, ktorý obsahoval 2147 snímkov pri celkovej úspešnosti 97.3 percenta.

3.3 Doplnkové postupy k riešeniu detekcie pruhov

Pri detekcii pomocou snímky z kamery vždy vznikajú ruchy v obraze, ktoré narušujú detekciu. Existuje veľa klasických metód extrakcie jazdných pruhov, napríklad podpora regiónov pomocou smeru prechodu a inverzie, metóda založená na perspektívnom mapovaní a mnoho detekčných algoritmov založených na gradiente, ktoré slúžia na detekciu jazdného pruhu. Ako už bolo povedané, v mnohých systémoch dostupnej detekcie jazdných pruhov sa často používa gradient obrazu alebo hrana. Tieto metódy potrebujú nízku výpočtovú silu a dobre zafarbené čiary jazdného pruhu. Avšak môžu byť ovplyvnené tieňom alebo poveternostnými podmienkami. Analýzu jazdných pruhov možno rozdeliť z viacerých hľadísk. Jedným z nich je analýza jazdných pruhov založená na modeloch tzv. the model-based lane detection, pri ktorej prvým doporučením je, aby cestná čiara korešpondovala s geometrickým modelom, čím sa získajú parametre modelu a následne daná čiara zapadá.

Viaceré vedecké práce sa zaoberali problematikou modelového riešenia, ako príklad možno uviesť prácu Geiger a kol. [32], kde bol použitý Bayesov klasifikátor na získanie modelu generovania pravdepodobnosti na základe pixelov povrchu cesty a navrhnutá funkcia pravdepodobnosti v kombinácii s charakteristickou trajektóriou vozidla a miznúcich bodov. Rozpoznávanie jazdných pruhov bolo získané komparatívnymi parametrami divergencie. Bosaghzadeh a kol.[33] vyriešili problém so získaním uhla predného obrazu pomocou metódy PCA. Detekcia jazdného pruhu založená na vlastnostiach the feature-based lane detection oddeľuje jazdný pruh od skutočnej cestnej scény na základe okraja a farebných vlastností jazdného pruhu. Zheng a kol. [34] transformovali pôvodný obraz RGB na CIE farebný priestor a na získanie efektívnych vlastností jazdného pruhu v cestnej scéne použili adaptívnu metódu dvojitého prahovania. Haselhoff a kol.[35] navrhli metódu detekcie ľavého a pravého pruhu pomocou dvojrozmerného lineárneho filtra, ktorý by mohol eliminovať

interferenčný šum počas detekčného procesu a zachovať inherentné vlastnosti jazdného pruhu aj z diaľky zobrazené na obr. 3.19.



Obr. 3.19: Výstup práce [35]

Son a kol.[36] sa zamerali na vplyv zmien svetla na detekciu jazdných pruhov. Na základe invariantnosti osvetlenia pruhov boli ako kandidátne pruhy vybrané žlté a biele pruhy a na detekciu bola použitá metóda zoskupovania clustering. Amini a kol. [37] pomocou Gaborovho filtra predpovedali únikový bod jazdnej dráhy v zornom poli. Keď bola funkcia spoľahlivosti na maximálnej úrovni spoľahlivosti, čiara generovaná pod úbežníkom bola hranicou jazdného pruhu.

V posledných rokoch sa niektorí odborníci a vedci zamerali najmä na neštruktúrované cesty a detekciu obrubníkov. Kong a kol.[38] použili algoritmus úbežného bodu na spätné zistenie hranice cesty a dosiahli dobré výsledky pri neštruktúrovanej detekcii cesty. Hervieu a kol.[39] použili na zistenie hranice cesty uhol medzi normálou roviny vybavenej údajmi o miestnych bodoch a normálom zeme a na predikciu hranice cesty bol použitý Kalmanov model filtra. Výsledky overenia ukázali, že metóda bola účinná pri detekcii hraníc ciest s prekážkami.

Kapitola 4

Vlastná implementácia

Pri prechádzaní jednotlivých riešení bolo potrebné zaviesť riešenia do praxe a skĺbiť výhody jednotlivých riešení. Kapitola sa zameriava na popis implementácie riešenia programu na detekciu jazdného pruhu v obraze. Obraz, ktorý program využíva je zachytený na kameru, prenesený do počítača a až následne sa spracúva. Toto rozhodnutie bolo uskutočnené z dôvodu technickej náročnosti problému. Program sám teda neumožňuje spracovanie videa v reálnom čase. Avšak program môže slúžiť ako predpríprava na túto činnosť. V kapitole sú taktiež popísané problémy a dostupné riešenia, ktoré sa vyskytli počas vývoja programu. Program je rozdenený na dve časti a to je časť, ktorá pozostáva z neurónovej siete a časť, ktorá využíva metódy bez učenia. Následne dochádza k zlúčeniu oboch riešení. Toto riešenie je ďalej testované a sú navrhnuté riešenia na jeho zlepšenie. Program by mal byť schopný analyzovať jazdný pruh a vykresliť ho na vozovke. Metódy, ktoré musí program v sebe implementovať pochádzajú z knižníc Tensorflow a OpenCV. Tieto knižnice boli zvolené z dôvodu veľkej podpory komunity dostupné riešeniam a dobrej zdokumentovanosti zdrojového kódu pre prípadne ďalšie rozširovanie programu. Ako bolo vyššie spomenuté, program je rozdelený do dvoch nezávislých častí, ktoré vykonávajú rôzne činnosti:

- **Časť bez učenia**: pri tejto metóde program využíva funkcie z knižnice OpenCV, ako sú napríklad Canny alebo Houghová transformácia tak, aby výsledkom tejto časti programu bol obrázok s vyznačenými jazdnými pruhmi na vozovke.
- **Časť s učením**: časť s učením sa zameriava na využitie neurónovej siete a prepojenie prvej časti s neurónovou sieťou, ktorá je natrénovaná na detekciu cesty. Pri behu programu sú využité funkcie z knižnice Tensorflow. Výstup programu je zobrazenie výseku jazdného pruhu.

4.1 Požiadavky na návrh programu

Jednou z hlavných častí pri vývoji programu je podstatne určenie očakávania alebo inak povedané ciele programu, ktoré by mal spĺňať svojou funkčnosťou. Definovanie týchto cieľov, požiadaviek by

malo byť dôležité v úvodnej fáze vývoja programu, na základe čoho bude program hodnotený. Pri definovaní požiadaviek je hlavné si funkcionalitu rozdeliť do bodov, ktoré môžu byť v priebehu času vyhodnocované ako splnené alebo nespnlené. Program, ktorý je vyvíjaný v rámci tejto diplomovej práce by mal spĺňať nasledovné body:

- **Jednoduchá použiteľnosť** - pre pužívateľa sú lákavejšie programy, ktoré možno neprinášajú vynikajúce výsledky, ale ich rozhranie alebo schopnosť obsluhy je natoľko jednoduchá, že sa pri práci používateľ cíti príjemne a komfortne. Preto by malo byť prvoradým cieľom čo najviac zjednodušiť používateľovi prístup k funkciám programu a dobre popísať ich obsluhu a zadávanie parametrov.

- **Nezávislosť od použitej kamery** - rozmanitosť trhu poskytuje množstvo výrobkov, ktoré sa svojimi parametrami od seba líšia. Diverzita však v tomto prípade predstavuje problém, s ktorým by sa mal program vysporiadať a fungovať naprieč rôznym použitým hardvérom na snímanie obrazu.

- **Jendouchá rozšíriteľnosť** - základné funkcie programu môžu byť využité pri iných projektoch, kde bude vyžadovaná odlišná funkcionalita, avšak postavená na už naprogramovaných funkciách. Preto umožnenie jednoduchého rozšírenia pomôže programu udržať sa vo svete detekcie jazdných pruhov aj v budúcnosti.

- **Štatistické údaje** Používateľ by mal byť schopný sledovať funkcionalitu programu v reálnom čase a hodnotiť prácu v programe počas jeho behu.

- **Prehľadné znázornenie výsledkov** - program sám bude schopný prevziať snímky cestných situácií a zobraziť používateľovi prehľadné vykreslenie výsledkov na danej snímke, čo umožní aj spätné overenie správnosti pre používateľa.

- **Kompatibilita v rámci OS** - dnešná doba umožňuje využívanie rôznych operačných systémov, ktoré sa od seba líšia, čo zapríčiňuje delenie trhu. Preto program, aby zasiahol čo najviac ľudí musí spĺňať požiadavku na fungovanie v rámci rôznych operačných systémov, ako je Windows, Linux alebo Mac OSX

4.2 Použité technológie

Práca na riešení pozostáva v druhom kroku po stanovení základných podmienok na program v správnom výbere technológií, ktoré budú použité na jeho vývoj tak, aby bolo čo najefektívnejšie dosiahnutie splnenia vyššie spomínaných podmienok. Prvý krok pri zvolení technológií je výber správneho programovacieho jazyka, za ktorým nasleduje výber správnej knižnice. V informatike sa pod pojmom knižnica označuje súbor funkcií alebo zdrojov, ktoré môžu zdieľať viaceré počítačové programy a často sú využívané pre vývoj softvéru. Ide o vopred napísaný kód a podprogramy, triedy,

hodnoty alebo špecifikácie typov. Knižnice umožňujú využitie už existujúcich kódov pre potreby tvorby nových zdrojových kódov v rôznych programoch. Služby knižníc sú poskytované prostredníctvom aplikačného rozhrania API (Application programming interface), na základe ktorého je určený spôsob, akým sa majú volať jednotlivé funkcie knižnice zo zdrojových kódov. Svetovo na riešenie problematiky akou sa zaoberá táto diplomová práca na detekciu jazdného pruhu sa využívajú dva hlavné programovacie jazyky: Python a C++. Oba tieto jazyky vznikli približne v podobnej dobe a súčasne sa nezávisle od seba vyvíjali. Preto v sebe kĺbia množstvo rozdielov a oba jazyky majú svoje výhody a nevýhody. Pri záverečnom hodnotení zmienených parametrov bol zvolený ako programovací jazyk Python a to pre jeho jednoduchosť, množstvo dostupných riešení, ale taktiež veľkej používateľskej podpore, čo výrazným spôsobom zjedodušuje a urýchľuje prácu programátora. Pri použití tohoto jazyka boli využité knižnice OpenCV a TensorFlow. OpenCV je možné použiť pre operačné systémy Linux, Windovs a Mac OS. Z tejto knižnice je dôležité vyzdvihnúť hlavne časť na spracovanie obrázkov a videa, ktorá je využívaná aj v programe a to pri predspracovaní a filtrovaní obrazu.

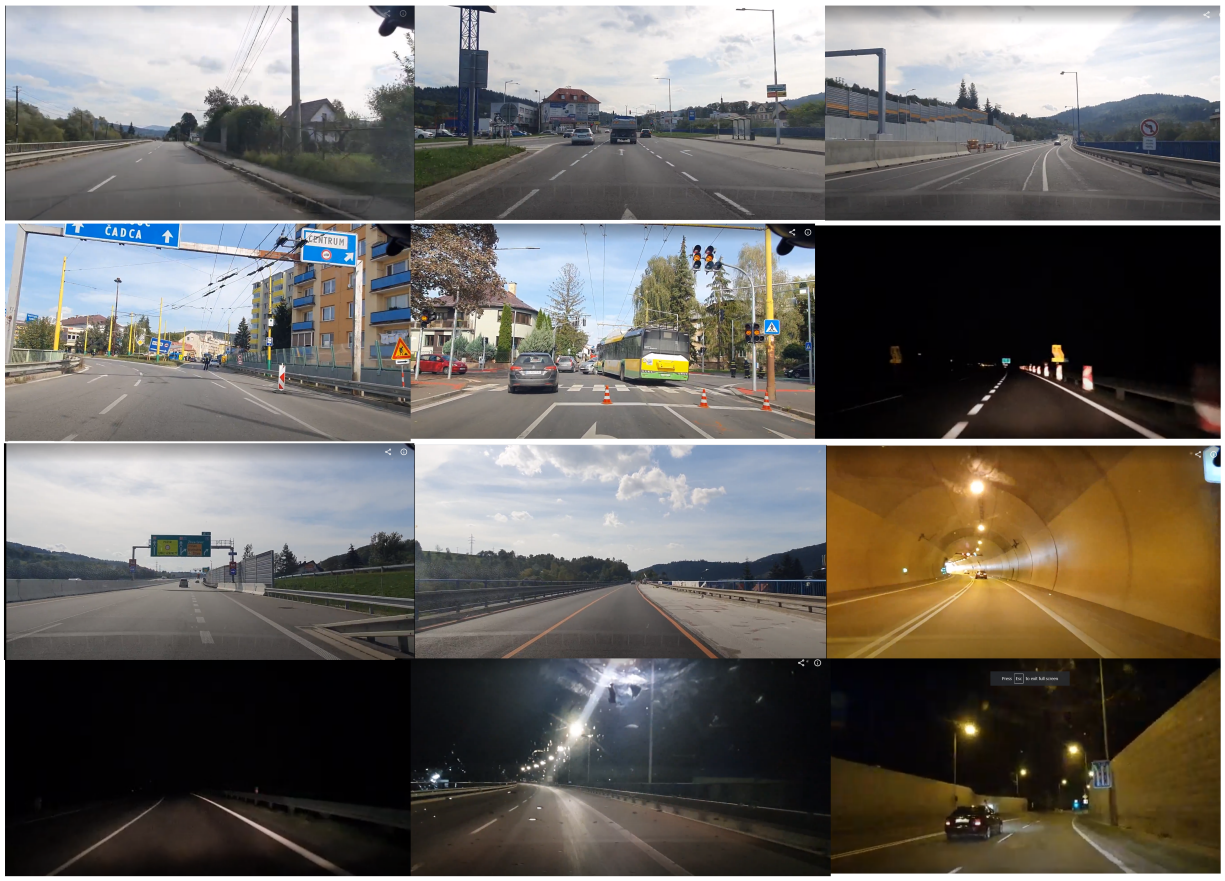
4.3 Vytvorenie sady cestných situácií

V tejto časti je rozoberaná problematika a vytváranie testovacích videí pre daný program. Jeden z menších cieľov bolo vytvorenie sady videí, ktoré sú lokalizované na cesty Strednej Európy. Preto boli videá natačané na úsekoch ciest I/647 Marianské Hory, cesta Sokolovská škola, ale tiež rýchlostná cesta D3 Svrčinovec-Skalité polovičný profil, kde boli natáčané nočné zábery. Ďalšou cestou je cesta I/11 smerom do Žiliny a taktiež mestský žilinský okruh. Aby bola zabezpečená rozmanitosť videí natáčali sa za rôznych svetelných podmienok. Pre dosiahnutie čo najlepšieho obrazu cesty boli kamery umiestnené takmer v strede čelného skla pod spätným zrkladom. Toto umiestnenie je podstatné z hľadiska natvrdo určeného regiónu záujmu, ktorý bude popísaný v podkapitole 4.4. Ako hlavné kamery boli zvolené kamery, ktoré obsahuje mobilný telefón Samsung S8 a Samsung S10. Kamery, ktoré sa dodávajú do automobilov od výrobcov neboli zvolené z dôvodu ich nedostupnosti. Avšak ich technické parametre boli podobné a v niektorých prípadoch boli parametre kamery Samsungu S8 a S10 lepšie ako parametre autokamery. Nočné zábery boli na kamerách S8 a S10 porovnateľne horšie ako na autokamerách z dôvodu chýbajúceho IR senzora. Parametre kamier :

	S10	**S8**
Rozlíšenie (foto)	4000 x 3000 Pixelov	4000 x 3000 Pixelov
Rozlíšenie(video)	3840x2160 @ 30 fps	1920x1080 @ 30 fps
Senzor	ISOCELL Plus	Exmor-RS CMOS
Clona	F1.9	F1.7

Tabuľka 4.1: Parametre použitých kamier

V sade cestných situácií ako je vidieť na obr. 4.1 sú zachytené: bežné cesty bez diaľníc, viacpruhové cesty v meste, novo zakryté cestné pruhy, tunely, radenie sa z pruhu do pruhu, veľký kruhový objazd, nehoda, diaľnica v noci s diaľkovými svetlami, diaľnica v noci bez diaľkových svetiel, mesto v noci, dážď v noci v meste.



Obr. 4.1: Sada cestných situácií

4.3.1 Kalibrácia kamery

Skupina detektorov, ktoré budú obraz spracovávať a filtrovať z neho nepotrebné informácie sú často závislé na pozícii kamery v aute. Napríklad, keby bola kamera umiestnená v pravom hornom rohu čelného skla, obraz by trpel značným vykrivením čiar a zdalo by sa, že stredová čiara, ktorá delí vozovku na dve časti prechádza stredom vozidla, čo by malo za následok pri detektore, ktorý skúma situáciu, vyhodnotenie ako vybočenie z pruhu. Preto je dobre si na úvod zadefinovať pozíciu kamery. Program sám bude počítať s touto pozíciou a teda objekty, ktoré bude detekovať ako správne čiary sa nebudú výrazne líšiť od reality. Ďalším aspektom pri kalibrácií kamery je nastavenie citlivosti na svetlo. Ak bude snímka preexponovaná, čiary na ceste nebude vôbec vidno, pretože

splynú s cestou. Moderné zariadenia, ako sú kamery mobilných telefónov obsahujú funkciu, ktorá automaticky nastaví expozíciu danej scény a tým uľahčí detekciu pri prechodoch z rôznych svetlných prostredí ako je, napríklad výjazd z tunela. Avšak aj tieto funkcie potrebujú svoj reakčný čas na aplikovanie odozvy na zmenu, čo spôsobuje dočasnú slepotu pre program.

4.4 Algoritmus bez učenia

Najrozšírenejšia a najviac dostupná metóda detekcia jazných pruhov je metóda, ktorá v sebe používa segmentáciu obrazu a snaží sa vytiahnuť a odfiltrovať nepotrebné informácie. Algoritmus, ktorý je popísaný nižšie, využíva tieto metódy tak, aby bolo dosiahnuté čo najlepšie vykreslenie cestnej čiary. Pri tomto algoritme sú používané funkcie z knižnice OpenCV ako napríklad Canny alebo GaussianBlur.

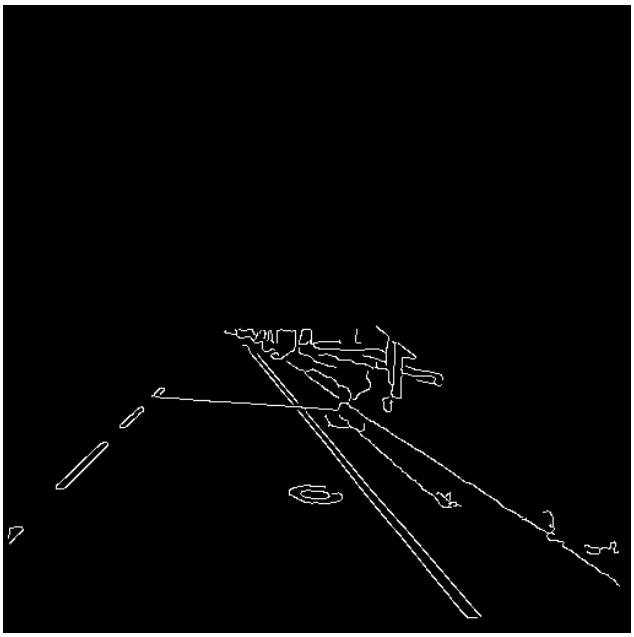
4.4.1 Predspracovanie

Aby bol algoritmus menej náročný na výpočetný výkon je potrebné, aby bolo z obrázka odstránených čo najviac dát, ktoré sú nepotrebné a boli spracovávané len tie dôležité. Prvý krok je preto určenie si regiónu záujmu. Keďže v podkapitole 4.3.1 bolo určené, že kamera má stabilné miesto v strede čelného skla pod spätným zrkadlom, je teda možné si na pevno zadefinovať tento výsek. Výsek je podobný lichobežníku a vyberá len časť cesty, ktorá bude ďalej sprcovaná. Aj keď kamera má presné určenie, výsek cesty je zvolený tak, aby bolo možné s kamerou mierne manipulovať. Druhým dôvodom pre mierne väčší výsek obrazu je parameter kamery a jej prirodzené priblíženie, čo by malo za následok zlý výber časti cesty. Druhým bodom pri predspracovaní obrázka je prevedenie obrázka do čiernobielej. Tým sú odstránené dve ďalšie dimenzie obrázka a dosahuje sa menšia výpočetná náročnosť. Na obrázku 4.2 je vidno cestnú situáciu po prevedení prvých dvoch krokov. Obrázok obsahuje fialovú výseč, ktorá zvýrazňuje vybranú časť cesty.

Obr. 4.2: Obrázok po úvodných krokoch

4.4.2 Detekcia čiar v obrázku

Aby bolo možné vykresliť čiary v obrázku je potrbená detekcia hrán. Ešte pred samotnou detekciou bola použitá funkcia GausianBlur, ktorá obrázok rozmaže tak, aby sa odstránili drobné ruchy v obraze. Ako funkcia pre detekciu hrán v obrázku bola zvolená funkcia Canny z OpenCV. Presný popis funkcionality funkcie Canny je popísaný v kapitole 3. Pri tejto funkcii je potrebné určiť hodnotu prahu, od ktorej sa bude odvíjať detekcia hrany. V tomto prípade bola zvolená hodnota (210, 255). Po prebehnutí funkcie je dosiahnutý čierny obraz, na ktorom sú vyznačené bielou farbou hrany, ktoré spĺňali zadané parametre.

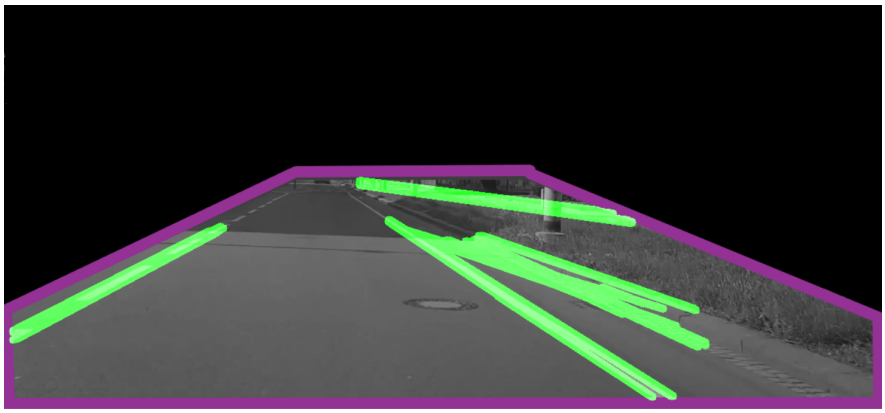


Obr. 4.3: Obraz po detektore Canny

Ako je možné vidieť na obr. 4.3 pomocou regiónu záujmu bolo dosiahnuté odfiltrovanie veľkého množstva ruchov v obraze, ktoré by sťažovali detekciu čiary. Avšak kvôli nie príliš presnému určeniu regiónu záujmu nebolo možné odstrániť všetky ruchy. Preto bola aplikovaná metóda prahovania, čo pomohlo odfiltrovať niektoré ruchy v obraze.

Obr. 4.4: Obraz po aplikovaní prahovania a Canny

Obrázok, ktorý bol na výstupe funkcie Canny je pripravený pre vykreslenie čiar. Na detekovanie čiar bola použitá Houghová tranformácia a nájdené body boli pospájané do čiar. Avšak ako je vidno na obr. 4.5 čiary, ktoré boli určené sú ovplyvnené ruchmi alebo zlým odfiltrovaním objektov pri určovaní regiónu záujmu.



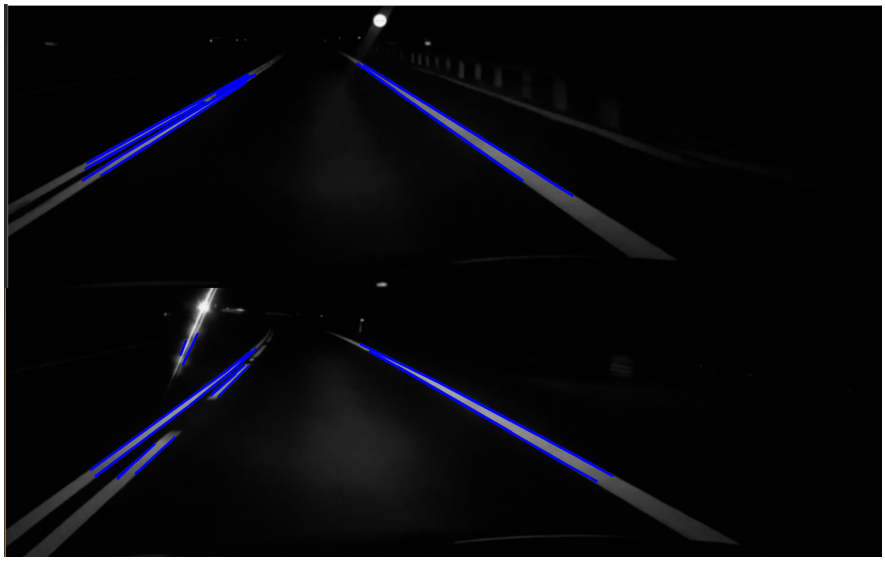
Obr. 4.5: Obraz po Houghovej tranformácií

4.4.3 Chyby vznikajúce pri detekcii

Počas behu programu je možné pozorovať nepresnosti detekcie jazdného pruhu. Tieto chyby alebo inak povedané chybné detekcie sa vyskytujú pri snímkach, kde je nadbytok objektov pri detekcii blízko cestných čiar. Základné aspekty ovplyvňujú chybovosť programu:

- **Nepresné určenie regiónu záujmu** - pri tomto probléme je región záujmu príliš veľký, čo má za následok zle odfiltrovanie objektov, ktoré nepatria možnej čiare, ako je možné vidieť na obrázku 4.5. Pri určovaní ROI bolo prihliadané na manipuláciu s kamerou a tým pádom musel byť región záujmu vačší. Objekty hlavne z kraja krajnice boli postúpené ďalšej analýze a vyhodnotené ako druh čiary. Tento problém je patrný aj pri osvetlenom obrubníku cesty, kedy dochádza k detekcii hrany a keďže detekovaná hrana ma rovnaké smerovanie ako hrany ciest, je táto hrana ďalej zobrazovaná.

- **Nočné zábery a oprotiidúce automobily** - kamera, ktorá sníma nočné zábery je častokrát automaticky prepnutá na vyššiu citlivosť svetla. Ak automobil, ktorého jazdný pruh je sledovaný ide po ceste v noci na okresnej ceste, kde nie je veľká intenzita vozidiel, je pre detektor jednoduché určiť pozíciu čiar a pekne ich vykresliť. Vtedy čiary na ceste majú vysokú saturáciu od reflektorov vozidla. Ak však ide oproti vozidlo, kamera umiestnená za čelným sklom vozidla vníma svetla vozidla ako odlesk od čelného skla, čo zapríčiňuje, že parametre odlesku sú častokrát zhodné so saturáciou aj smerovaním čiary na ceste, avšak sú lokalizované v inej pozícii, čo je možné vidieť na obr. 4.6



Obr. 4.6: Chybná detekcia pri odleskoch svetla

- **Svetelnosť scény** - pri prechodoch rôznymi svetelnými prostrediami sa kamera prispôsobuje prichádzajúcemu svetlu. Pri vjazde alebo výjazde z tunela je možné, že scéna bude preexponovaná a prahovanie, ktoré bolo zvolené odstráni z obrázka aj tie aspekty, ktoré boli potrebné. Pri ďalších krokoch vzniká naopak problém s nedetekovaním pruhu. Tento problém je možné pozorovať aj pri zmene farby čiar na ceste, kde čiary ktoré slúžia ako obchádzkové dočasné značenie majú častokrát oranžovú alebo červenú farbu. Pre detektor nedosahujú potrebnú saturáciu a tak ich detektor odstráni .

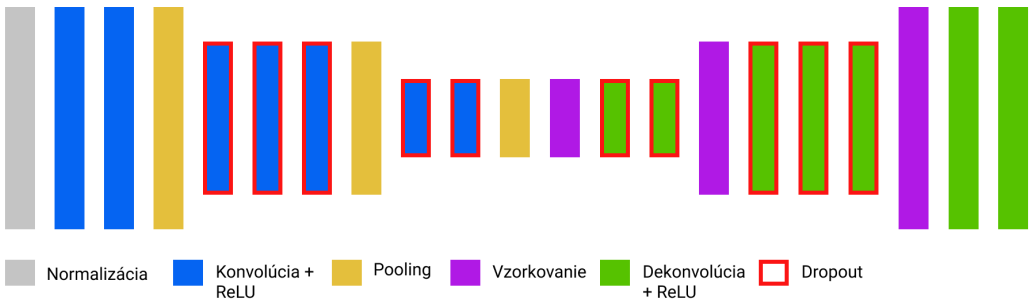
4.5 Určenie RIO pomocou neurónovej siete

Vačšina problémov, ktoré boli popísané v podkapitole 4.4.3 je spôsobená zlým alebo príliš veľkým určením regiónom záujmu. Keďže každý automobil je iný a má iné parametre, ako napríklad je svetla výška od vozovky, šírka automobilu alebo veľkosť senzorov pri spätnom zrkadle. Nedá sa na pevno určiť poloha kamery tak, aby spĺňali parametre pre správnu detekciu. Preto bola vybraná

neurónová sieť, ktorá sama určí región záujmu a výsledné snímky ďalej poskytne pre detektor na ďalšie spracovanie a určenie jazdného pruhu.

4.5.1 Štruktúra neurónovej siete

Pri vyberaní vhodnej neurónovej siete bola zvolená neurónová sieť od autora Michael Virgo [1]. Táto sieť je plne konvolučná neurónová sieť pre detekciu cesty. Teda je možné povedať, že nedetekuje priamo jazdné pruhy, čo však pre účely, na ktoré je využitá v tejto diplomovej práci úplne postačuje. Sieť využíva funkcie z knižnice keras. Vrstvy siete sú nastavené tak, aby sa postupne zmenšovala ich veľkosť a následne nasleduje dekonvolúcia. Autor sa rozhodol vytvoriť svoj model ako zrkadlový obrazom konvolučných vrstiev . Konvolučné vrstvy sa postupne zmenšujú až do stredového bodu, kedy nastáva vzorkovanie a dekonvolúcia tak, aby bola každá vrstva kópiou vrstvy, ktorá je rovnako vzdialená od stredového bodu v opačnom smere. Posledná výstupná vrstva končí jedným filtrom. Tento filter slúži na vrátenie obrázka s hodnotami len v kanali Gray, kde obsahuje údaje o vyznačenej ceste. Neurónová sieť prijíma obraz v rozlíšení 160x80 s troma kanálmi pre farebný obrázok. Tento rozmer bol zvolený pri dosahovaní kompromisu medzi pamäťovou náročnosťou a presnosťou detekcie. Farebné aspekty boli zachované pre lepšiu detekciu farebných čiar na ceste. Výsledný model je v súbore fully_conv_NN.py. Na obr. 4.7 je zobrazený model konvolučnej neurónovej siete, ktorý sa využíva pri riešení problému analýzy jazdných pruhov.

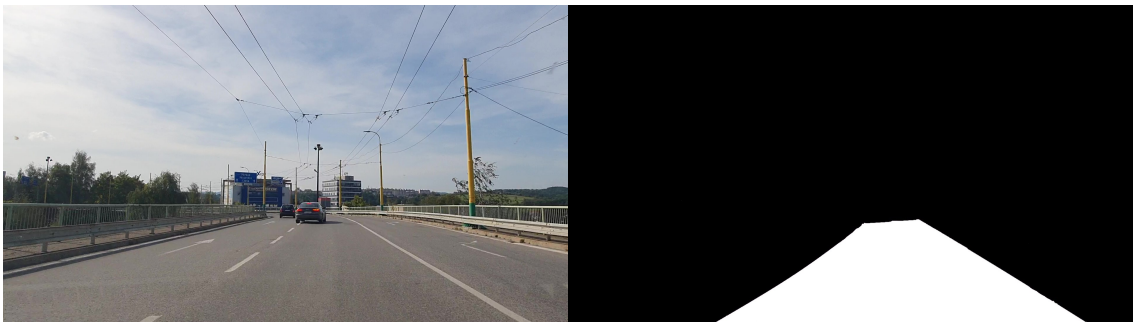


Obr. 4.7: Model konvolučnej neurónovej siete

4.5.2 Trénovacie dáta

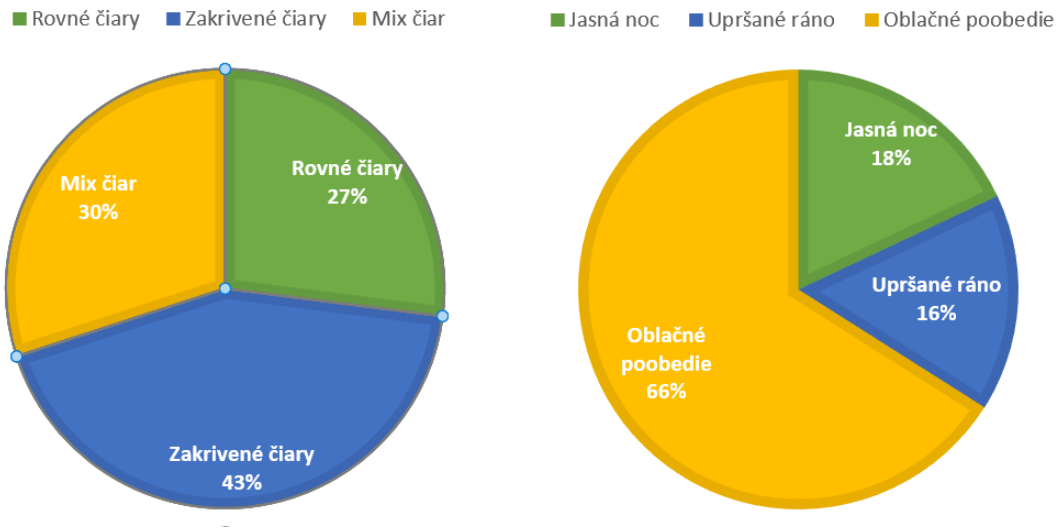
Sieť, ako bolo vyššie spomenuté, prijíma na vstupe obrázok v rozlíšení 160x80 s troma kanálmi (farebný obrázok). Preto autor sám vytvoril dataset a natrénoval neurónovú sieť. Autorov dataset sa skadá vždy z dvoch obrázkov v rozlíšení 168x80 z toho jeden je cestná situácia. Tento obrázok je farebný a druhý obrázok je s anotovanými dátami, ktorý popisuje pre neurónovú sieť správnosť výstupu. Tento obrázok obsahuje len jednu dimenziu. Príklad je možný vidieť na obr. 4.8

[1] https://github.com/mvirgo/MLND-Capstone



Obr. 4.8: Príklad datasetu pre neurónovú sieť

Autorova sada cestných situácií sa skladá z kombinácie rôznych počasí a rôzneho zakrivenia cestných čiar. Dataset obsahuje rovné čiary, zakrivené čiary , oblačné poobedie, jasnú noc, upršané ráno. Percentuálne rozdelenie datasetu je možné vidieť na obr. 4.9.



Obr. 4.9: Graf zobrazujúci rozmanitosť datasetu

Pre potreby pretrénovania bol vytvorený vlastný dataset, ktorý bol natočený na lokálnych cestách, avšak tento dataset nemal takú rozmanitosť a veľkosť ako dataset autora. Dataset pre účely diplomovej práce sa skladá z 200 cestných situácií v rozlíšení 1080p a 30 fps. Oproti tomu dataset autora mal 11420 rôznych cestných situácií, avšak autor natáčal snímky ciest na rozlíšenie 720p a 30fps. Vytvorený dataset bol natrénovaný na neurónovej sieti, avšak jej presnosť bola výrazne horšia, preto bolo rozhodnuté ponechať pôvodnú sieť. Táto sieť mala však zhoršenú detekciu cesty v tuneloch. Preto bola sieť znova natrénovaná na pôvodných dátach, ale bolo využité viac epoch, konkrétne 100 epoch od pôvodných dvadsiatich. Vtedy sa hodnota stratovej funkcie pohybovala okolo 0,0043. Táto natrénovaná sieť bola použitá v ďalšom postupe.

4.5.3 Generované výstupy neurónovou sieťou

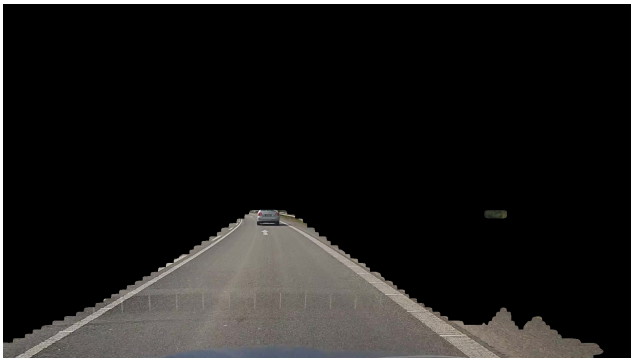
Aby bola neurónová sieť vhodná na použitie v diplomovej práci generuje len malú masku o rozmeroch 160x80. Maska predstavuje vyznačenú cestu, alebo lepšie povedané predikciu neurónovej siete o umiestnení cesty v obrázku. Táto predikcia je používaná v ďalšej fáze a je opísaná v podkapitole 4.6

Obr. 4.10: Vystup neurónovej siete

Na obr. 4.10 je možné vidieť výstup neurónovej siete, ktorý je len čiernobiely obrázok,kde bielou farbou je vyznačená cesta. Pri detekcii neurónovej siete vznikajú občas chyby, ktoré sa prejavujú ako čierne diery v ceste, preto program sa snaží tieto objekty vyplniť ešte pri tvorení masky.

4.6 Finalizácia a výstup riešenia

V predchádzajúcich krokoch boli popísané prístupy k detekcii jazdných pruhov. Pre ich zdokonalenie bol vyskúšaný postup zlúčenia týchto prístupov. Po úspešnom určení regiónu záujmu je potrebné masku, ktorá bola obdržaná z neurónovej siete zväčšiť a aplikovať na obrázok. Po prevedení tohoto kroku je možné vidieť vystup na obr. 4.11.



Obr. 4.11: Výstup po aplikacií regiónu záujmu

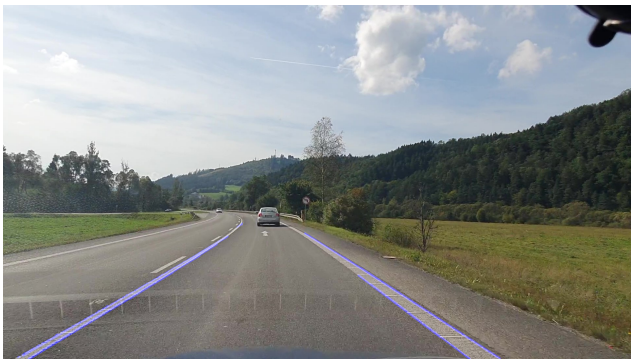
Potom sú na obrázok aplikované metódy ako Canny a Houghová transformácia, čím je dosiahnutá extrakcia čiar z obrázka. Avšak pri detekcii vznikali chyby ako vodorovné čiary, napríklad pri zlomoch betónových úsekoch na diaľnici, alebo odledky od kapoty, ktoré majú na obrázku vodorovný parameter. Túto chybu je možné vidieť na obr. 4.12. Preto je aplikovná funkcia na odstránenie vodorovných čiar, to je možné vidieť na výpise 4.1.

Obr. 4.12: Chyba pri detekcií vodorovných čiar

```
def _is_line_horizontal(self, p1, p2):
      y_diff = np.abs(p1[1] - p2[1])
      return y_diff < 50
```

Výpis 4.1: Odstránenie vodorvných čiar z obrázka

Po aplikácii týchto výstupov sú do obrázka vykreslené čiary, ktoré našiel detektor. Obr. 4.13 zobrazuje výstup navrhnutého riešenia, ktoré bolo aplikované na obrázok lokálnej cesty.



Obr. 4.13: Aplikácia programu na lokálnu cestu

Program zobrazuje nájdené čiary modrou farbou. Pre ich nepoužívanie v cestných situáciách, kde sa z väčšej časti používajú biele a oranžové farby. Program bol testovaný na videách, ktoré boli zhotovené z lokálnych ciest. Príkladové testovacie video na nachádza v prílohe pod názvom test.mp4. Video je v rozlíšeni 1920x1080.

Kapitola 5

Testovanie navrhnutého riešenia

Táto kapitola je zameraná na otestovanie navrhnutého riešenia. Preberá postup testovania, zvolené testovacie dáta a taktiež ich popis. Výsledky sú zobrazené v tabuľke. Kapitola sa zaoberá určením dôvodu chybovosti na jednotlivých obrázkoch.

5.1 Princíp testovania

Program na testovanie je umiestnený v súbore test.py. Na úvod programu je potrebné nastaviť odkiaľ sa dáta budú čerpať, kde sa nachádzajú testovacie obrázky a kde sú anotované dáta. Úvodný program obsahuje možnosť otestovať ľubovoľný obrázok v súbore, ale tiež dovoľuje otestovať všetky obrázky v priečinku. V tejto časti sa bude opisovať postup pri načítaní jedného obrázka zo súboru, pretože postup pri viacerých obrázkoch je rovnaký. Program načíta obrázok z vybraného súboru. Obrázok odošle do neurónovej siete, aby získal predikciu cesty. V podkapitole 5.2, kde sú rozoberané testovacie dáta bola zvolená ako farba čiar ružová, preto je potrebné ju konvertovať do čiernobielej. Bol zvolený určitý rozsah farby, ktorá je prijateľná ako farba čiary. V tomto bode je možné pristúpiť k počítaniu čiar v obrázku, ktorý našla neurónová sieť, aby sa predišlo pádom programu. Program zistí, či detektor nejakú čiaru našiel a ak nie, tak overí, či je to správny výsledok, to je možné vidieť na výpise kódu 5.1.

```
if white_count == 0:
            gt_white_count = cv2.countNonZero(gt_bw)
            if gt_white_count > 0:
                return 0.0
```

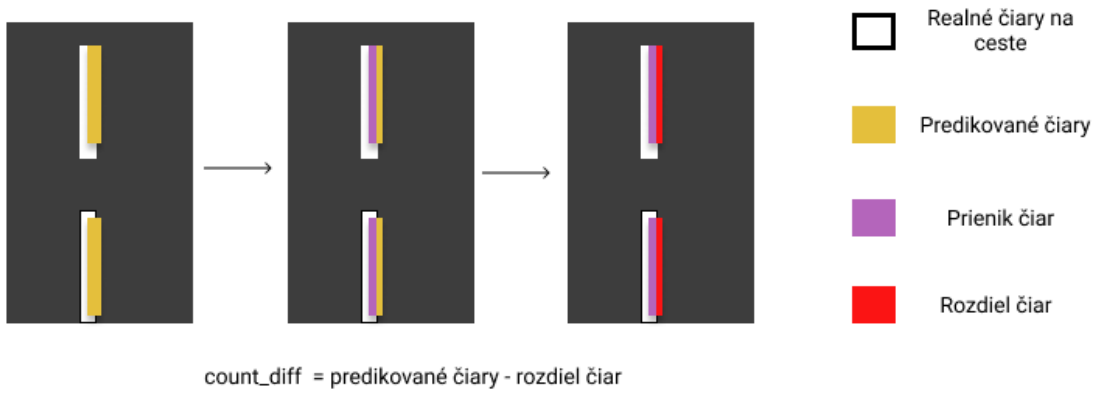
Výpis 5.1: Testovanie obrázka detekovaných čiar

Potom program pokračuje tým, že prekryje obe sady čiar cez seba a odstráni čiary, ktoré sú predpísané v anotovaných dátach. Pri tomto bode dostane program obrázok obsahujúci čiary, ktoré boli

zle detekované. Úspešnosť detekcie sa počíta podľa nasledujúceho vzorca:

$$accuracy = \frac{count\_diff * 100}{white\_count} \tag{5.1}$$

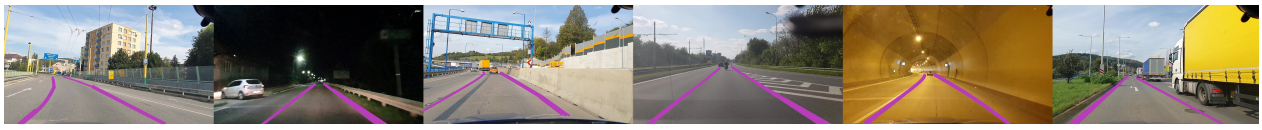
kde white_count - všetky detekované čiary algoritmom na detekciu a count_deff - rozdiel medzi detekovanými čiarami algoritmu a rozdielom prieniku medzi čiarami na ceste a výsledkom detektora. Pre lepšiu predstavu je postup zobrazený na obr. 5.1.



Obr. 5.1: Postup algoritmu na testovanie

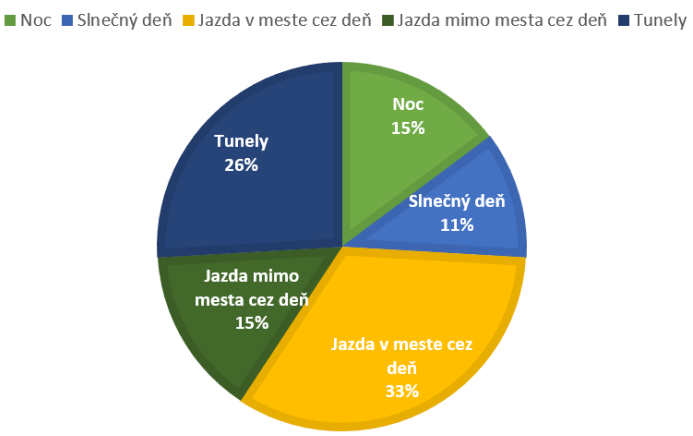
5.2 Testovacie dáta

Možnosť čo najjednoduchšieho pridávania a overovania výsledkov bola aplikovaná pri tvorbe datasetu na testovanie zvoleného riešenia. Pre tento dôvod bola zvolená ako farba pre anotovanie ružová, konkrétne v RGB rozsahu (156, 48, 152) až (201, 79, 197).



Obr. 5.2: Anotované dáta pre testovanie

Pri tejto farbe stačí anotované dáta len vyznačiť a ostatné farby ponechať na obrázku. V programe na testovanie sa ostatné farby odstránia a program pracuje už len v čierno bielych hodnotách. Obr. 5.2 sú zobrazené príklady z anotovaných dát z testovacieho datasetu. Aby bola docielená rozmanitosť dát, základná testovacia sada je tvorená 54 obrázkami z rôznych situácií.



Obr. 5.3: Graf zobrazujúci percentualne zastúpenie datasetu

Na obr. 5.3 sú zobrazené percentuálne zastúpenia jednotlivých cestných situácií v datasete na testovanie. Na obrázku sa taktiež nachádza údaj slnečný deň. Tento údaj znamená situáciu, kedy slnečné lúče buď priamo alebo odrazom zachytávala kamera a na snímkach vznikal dojem prelínania farieb medzi cestou a čiarou, čo sťažovalo detekciu.

5.3 Výsledky testovania

Na úvod testovania bola otestovaná rýchlosť programu. Program bol spustený na počítači s procesorom Intel Core i7-8750H s taktom 2.20GHz a grafickou kartou Nvidia Geforce GTX 1050Ti. Počítač spracovával video, ktoré bolo natočené v rozlíšení 1920x1080 s 30 fps. Išlo o 20 sekundové video. Počítaču trvalo 51 sekúnd spracovať video, čo je 11.8 snímkov za sekundu. Druhým testovacím počítačom bol počítač s procesorom Intel Core i5-4670 s taktom 3.4GHz a grafickou kartou Nvidia Geforce GTX 660. Spracovanie rovnakého videa trvalo 62 sekúnd, čo je v priemere 9.8 snímkov za sekundu.

Samotné testovanie najskôr prebiehalo na testovaní porgramu s použitou neurónovou sieťou a bez použitia neurónovej siete a teda natvrdo určenému regiónu záujmu. Výsledky týchto dvoch metód je možné vidieť v tabuľke 5.1.

	Základné parametre	**Upravené parametre**
Metóda s neurónovou sieťou	55.02%	70.10%
Metóda bez neurónovej siete	43.21%	39.82%

Tabuľka 5.1: Výsledky testovania jazdných pruhov metódou bez neurónovej siete a metodou s neurónovou sieťou

Výsledky pri metóde bez neurónovej siete boli výrazne horšie ako pri variante s neurónovou sieťou. V obraze vznikalo množstvo ruchov, ktoré neboli odfiltrované regiónom záujmu a výsledok obsahoval mnoho chybne detekovaných čiar. Toto je vidieť na obr. 5.4.

Obr. 5.4: Výstupný obrázok z metódy bez použitia neurónovej siete

Pri testovaní boli vytvorené rôzne druhy testovacích sád pre jednotlivé prostredia, kde sa menili parametre rôznych funkcií. Hlavne parametre Houghovej transformácie a Canny detektor hrán. Tieto zmeny sú poznačené v programe a taktiež aj v tabuľke 5.2

	Základné parametre	**Upravené parametre**
Noc	58.11%	65.06%
Slnečný deň	32.60%	74.90%
Jazda v meste cez deň	89.34%	87.91%
Jazda mimo mesta cez deň	77.01%	87.96%
Tunely	26.59%	62.35%
Priemer	55.02%	70.10%

Tabuľka 5.2: Výsledky testovania jazdných pruhov

Ako je možné vidieť v tejto tabuľke, úprava parametrov výrazne prospela vo väčšine situácií, avšak pri bežných denných snímkach došlo k malému zhoršeniu, čo sa však v konečnom dôsledku ukázalo ako správne pri výrazne zvýšenom percente všetkých dát dohromady. Zmenené parametre sú zobrazené vo výpise kódu 5.2.

```
canny_image = cv2.Canny(cutted, 210, 255, apertureSize=3, L2gradient=True)
lines = cv2.HoughLinesP(img, rho=1, theta=np.pi/180, threshold=100,
                        lines=np.array([]), minLineLength=100, maxLineGap=70)
#Upravené parametre
```

```
canny_image = cv2.Canny(cutted, 100, 255, apertureSize=3, L2gradient=True)
lines = cv2.HoughLinesP(img, rho=1, theta=np.pi/135, threshold=135,
                        lines=np.array([]), minLineLength=60, maxLineGap=15)
```

Výpis 5.2: Upravené parametre programu

Nočné zábery sú na pár obrázkoch poškodené dažďom, ktorý znižuje detekciu, taktiež tieto obrázky nie sú výrazne kvalitné z dôvodu zlej prispôsobivosti kamery nočnému svetlu. Veľký posun je zaznamenaný na porovnaní testovania slnečného dňa. Pri zmene parametrov došlo k výraznému zlepšeniu, na čom mala zásluhu zmena parametrov Cannyho detekcie hrán, kde detektor dokázal odlíšiť hrany napriek zhoršenej farebnej viditeľnosti a saturácii snímky. Tunelove snímky mali podobné problémy ako snímky slnečného dňa. Pre zlepšenie detekcie bola neurónová sieť pretrénovaná na tunelové scény, ktoré boli extrahované z videa z lokálnych ciest. Tento dataset bol natrénovaný na 100 epochách, ale nepriniesol požadovaný efekt. Stratová funkcia dosiala hodnotu 0,025. Pri testovaní na náhodnom videu bola úspešnosť detekcie výrazne horšia. Výstupný obrázok cesty obsahoval ruchy a slepé miesta kvôli čomu nebolo možné detekovať čiary.

Obr. 5.5: Výstup po pretrénovaní CNN na lokálnom datasete

Na obr. 5.5 je vidno slepé miesta - čierne pruhy prebiehajúce cez cestu, čo sťažuje detekciu pruhov.

5.3.1 Zmena veľkosti vstupného obrázka

Ako bolo spomenuté v kapitole 5.3 program sám nedokáže fungovať v reálnom čase. Tento nedostatok sa bol pokúšaný odstrániť zmenšením vstupných obrázkov. Obrázky boli zmenšené o polovicu. Program pri spracovaní obrázkov dosahoval oveľa rýchlejšie spracovanie obrázka, avšak bol výrazný rozdiel v kvalite výstupu. V tabuľke 5.3 sú zachytené testy, ktoré boli prevedené na rovnakom datasete. Tento dataset bol zmenšený na polovičnú veľkosť rozlíšenia.

	Základné parametre	Upravené parametre
Noc	40.08%	55.57%
Slnečný deň	29.05%	52.97%
Jazda v meste cez deň	50.12%	61.22%
Jazda mimo mesta cez deň	52.32%	58.37%
Tunely	24.15%	47.56%
Priemer	37.92%	54.48%

Tabuľka 5.3: Výsledky testovania jazdných pruhov pri rozlíšeni 960x540

Výsledok výstupu programu bol podstatne horší vo všetkých aspektoch oproti snímkam v rozlíšení 1920x1080. Toto rozlíšenie nie je príliš vhodné na použitie vo vytvorenom programe.

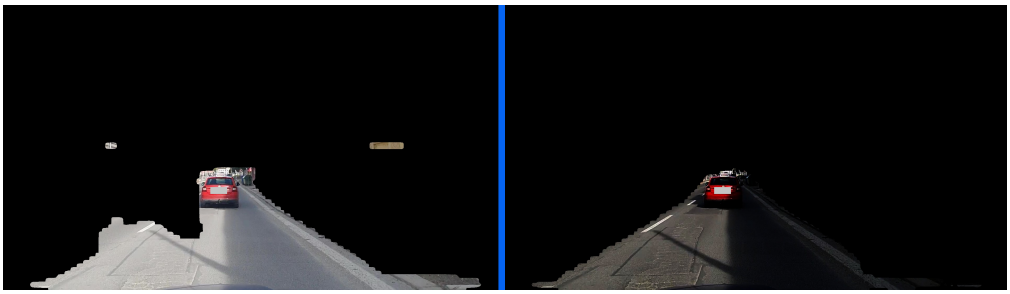
5.3.2 Gamma korekcia a prahovanie

Jas snímok pri ostrom dennom svetle hlavne počas slnečného dňa bol príliš vysoký. Z predchádzajúceho experimentu v tabuľke 5.2 je vidieť, že úspešnosť detekcie dosahovala 74.9 % pri slnečnom dni. Pre zlepšenie výsledku bola na obrázky použitá gama korekcia s hodnotami od 0,4 do 1,0. Tento postup nepriniesol výrazné zlepšenie, keď bol použitý priamo na výstup neurónovej siete. Percentá úspešnosti boli rovnaké, v niektorých prípadoch dosiahol výstup nižšie percento úspešnosti. V tabuľke 5.4 je možné vidieť zhoršenie detekcie pri použití parametra gamma 0,4. Snímka ešte obsahovala veľa nežiadúcich tmavých objektov, preto bolo použité prahovanie. Funkcia z OpenCV THRESH_TOZERO začierni len tie pixely obrázka, ktoré nevyhovujú parametrom a ostatné pixely ponechá v pôvodnom stave. Nastavené parametre prahovania boli (30,255).

	Uspešnosť
Bez použitia	74.90%
Gamma	74.10%
Gamma + Prahovanie	78.76%
Prahovanie	80.03%

Tabuľka 5.4: Výsledky testovania pri prahovaní a gamma korekcii

Avšak keď bola gamma korekcia použitá ešte pred vstupom obrázka do neurónovej siete, výstupný obrázok mal lepšie detekovanú časť cesty a tým bolo možné presnejšie určiť čiary. Pri predspracovaní bola použitá hodnota gamma korekcie 0,4. Táto hodnota vychádzala zo skúšania zmeny rôznych parametrov a vplyvu na detekciu. Na obr. 5.6 je na ľavej strane zobrazený výstup neurónovej siete bez použitia gamma korekcie a na pravej strane obrázka je zobrazený výstup neurónovej siete s použitím gamma korekcie.



Obr. 5.6: Výstup neurónovej siete bez použitia a s použitím gamma korekcie

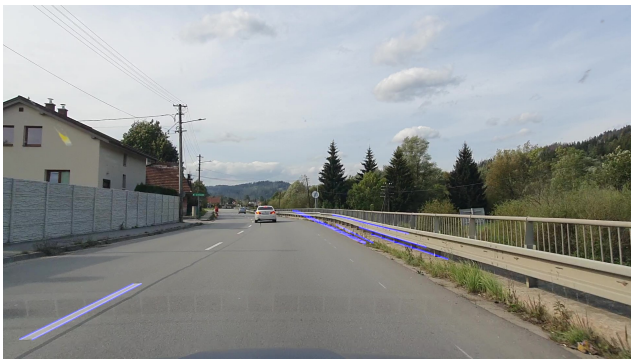
Táto príprava obrázka na spracovanie neúronovou sieťou výrazne zlepšila detekciu naprieč všetkými prostrediami, čo je možné vidieť v tabuľke 5.5. Postup prahhovania bol rovnaký ako v predchádzajúcom kroku a to až po výstupe z neurónovej siete.

	Uspešnosť
Bez použitia predspracovania CNN	70.10%
Gamma predspracovanie CNN	75.04%
Gamma predspracovanie CNN + Prahovanie	74.02%
Prahovanie	70.54%

Tabuľka 5.5: Priemerná úspešnosť detekcie s predspracovaním obrázka pre CNN

5.3.3 Odhalené chyby pri testovaní

Pri obdržaní niektorých výsledkov z neurónovej siete došlo k zníženiu percenta úspešnosti. Toto zníženie býva zväčša spôsobené zamenením si objektov v detektore ako napríklad, identifikovanie zvodidla. Pri niektorých prípadoch, hlavne počas jazdy cez dedinu sa program intuitívne snaží zobraziť pravú krajnicu ako čiaru. Je to v prípadoch, že pravá čiara na ceste chýba. Táto situácia je vyznačená programom ako chyba a snímky častokrát dosahujú iba 40% až 50% úspešnosť. Napríklad pri testovaní obrázka 24 v testovacom datasete obr. 5.7 je možné si všimnúť, že program správne detekoval jazdnú čiaru na ľavej strane, avšak keď chcel detekovať jazdný pruh sprava, najbližšia čiara bola tvorená zvodidlom cesty, preto identifikoval zvodidlo ako čiaru a dosiahol úspešnosť detekcie len 32% .



Obr. 5.7: Chyba programu na snímke 24

Problém taktiež vznikal v prípade, že automobil išiel proti svetlu. Cestné čiary splývali s vozovkou a nebolo možné správne určiť jazdný pruh. Tento problém však bol čiastočne odstránený zmenou kontrastu snímky a úpravou paramentrov jednotlivých funkcií. Ďalším prípadom, kedy detektor chybne detekoval zvolenú scénu bol prípad scény, kedy na ľavom okraji boli betónové zábrany. Vtedy sa podarilo detektoru detekovať deliace čiary jednotlivých betónových blokov. Táto detekcia vznikala, keď slnko svietilo za automobilom a od hrany sa odrážalo svetlo. Na výstupnom obraze vznikla vertikálna čiara, ako je možné vidieť na obr. 5.8.

Obr. 5.8: Chyba programu na zábrane

Kapitola 6

Záver

V súčasnosti sa technický pokrok ľudstva zrýchľuje. Čím ďalej, tým viac bežných vecí presúvame na roboty alebo automatizované systémy. V posledných rokoch zaznamenal veľký úspech sektor autonómnych vozidiel, ktoré dokážu dopraviť ľudí na miesto určenia. Automobily v sebe zahŕňajú množstvo použitých technológií a vedomostí. Tieto vedomosti si muselo ľudstvo časom získať a osvojiť. Výroba a vývoj takýchto vozidiel je v súčasnej dobe veľmi nákladná. Či sa už jedná o hardvérovú výbavu alebo najatý zástup inžinierov, ktorí sa podieľali na vývoji. Pri vývoji sa musí v prvom rade dbať na bezpečnosť posádky automobilu. Vytvoriť nástroj, ktorý bude detekovať zvolenú cestu a tým umožní bezpečný chod automobilu po vyznačenej trase za nízku cenu, je úlohou veľkých firiem. V súčasnej dobe sa na zachytenie obrazu cestnej situácie využívajú sady senzorov, ktoré automobil obsahuje. Tieto takzvané oči automobilu dokážu vytvoriť virtuálny obraz, kde stredobodom je automobil a pomáhajú monitorovať situáciu okolo vozidla.

Táto diplomová práca sa snažila využiť len jeden z týchto senzorov a to čelnú kameru umiestnenú pod spätným zrkadlom vodiča. Úvod práce sa zaoberal rôznymi situáciami, aké je možné v premávke pozorovať, hlavne čo sa týka cestných čiar. Tento problém bol analyzovaný z viacerých uhlov, preto v neskorších kapitolách sú spomenuté už vytvorené nástroje a riešenia, ktoré sa zoberali problematikou analýzy jazdného pruhu. V ďalšej časti sa diplomová práca zaoberá vlastnou implementáciou riešenia problému. Pri tomto bode bolo dbané na to, aby bolo možné využiť nástroj na rôznych druhoch vozidla, preto bola pozícia kamery v rámci vozidla pohyblivá. Neurónové siete sú populárne a vykazujú dobré výsledky pri analýze obrazu, čo bolo zistené v prácach, ktoré sa zaoberali touto problematikou. Práca sa snažila skĺbiť dva modely riešení a to metódu s trénovaním neurónovej siete a metódu bez trénovania, alebo bez použitia neurónovej siete, kde sa využívali rôzne postupy filtrovania obrazu. Navrhnutý program využíval neurónovú sieť ako svoj región záujmu a naň aplikoval filtrovacie metódy. Výsledný program bol otestovaný na lokálnom datasete, ktorý bol pre tento účel vytvorený.

	Základné parametre	**Upravené parametre**
1920x1080 rozlíšenie	55.02%	70.10%
960x540 rozlíšenie	37.92%	54.48%
Predspracovanie	61.24%	75.04%
Metóda bez neurónovej siete	43.21%	39.82%

Tabuľka 6.1: Skrátené výsledky testovania

Ako je možné vidieť v tabuľke 6.1 úspešnosť sa v mnohých aspektoch líšila. Pri testovaní bolo dbané aj na rýchlosť vytvoreného riešenia a boli analyzované zmeny parametrov alebo rozlíšení vstupných obrázkov. Testovaním boli zistené nedostatky v programe a pri jeho úprave sa podarilo zvýšiť úspešnosť riešenia o 15 %. Najlepšie výsledky program dosahoval vtedy, keď bolo vyhovujúce počasie - polooblačno bez ostrého svetla. Taktiež veľká úspešnosť detekcie bola zaznamenaná pri snímkach v nočnom prostredí bez protiidúcich automobilov. Aby bola úspešnosť detekcie ešte vyššia, bol zvolený postup predspracovania vstupného obrázka do neurónovej siete. Postup priniesol zlepšenie detekcie o 5% oproti testovaniu bez tohoto kroku.

Hlavným cieľom práce bolo preskúmať možnosti detekcie pruhov a overiť si ich na vytvorenom riešení a reálnej cestnej situácii, čo dokázalo že dostupné riešenia majú potenciál v použití aj na lokálnych cestách. Toto dokázali využité experimenty. Zvolené algoritmy by sa dali vylepšiť použitím lepších a dostupných kamier, ktoré majú dobrú citlivosť na svetlo a sú prispôsobené na využívanie v cestnej premávke.

Literatura

1. CAO, Jingwei; SONG, Chuanxue; SONG, Shixin; XIAO, Feng; PENG, Silun. Lane detection algorithm for intelligent vehicles in complex road conditions and dynamic environments. *Sensors*. 2019, roč. 19, č. 14, s. 3166.
2. NARAYAN, Aparajit; TUCI, Elio; LABROSSE, Frederic; ALKILABI, Muhanad H Mohammed. Road detection using convolutional neural networks. In: *Artificial Life Conference Proceedings 14*. 2017, s. 314–321.
3. DASHKO, Leonid. Road Detection and Recognition from Monocular Images Using Neural Networks. [B.r.].
4. OBRACZKA, Katia; MANDUCHI, Roberto; GARCIA-LUNA-AVECES, JJ. Managing the information flow in visual sensor networks. In: *The 5th International Symposium on Wireless Personal Multimedia Communications*. 2002, zv. 3, s. 1177–1181.
5. AKDERE, Mert; ÇETINTEMEL, Uğur; CRISPELL, Daniel; JANNOTTI, John; MAO, Jie; TAUBIN, Gabriel. Data-centric visual sensor networks for 3D sensing. In: *International conference on GeoSensor Networks*. 2006, s. 131–150.
6. WU, Youfu; CHEN, Zusheng. A detection method of road traffic sign based on inverse perspective transform. In: *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*. 2016, s. 293–296.
7. DAIGAVANE, Prema M; BAJAJ, Preeti R. Road lane detection with improved canny edges using ant colony optimization. In: *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. 2010, s. 76–80.
8. RONG, Weibin; LI, Zhanjing; ZHANG, Wei; SUN, Lining. An improved CANNY edge detection algorithm. In: *2014 IEEE international conference on mechatronics and automation*. 2014, s. 577–582.
9. XUAN, Li; HONG, Zhang. An improved canny edge detection algorithm. In: *2017 8th IEEE international conference on software engineering and service science (ICSESS)*. 2017, s. 275–278.

10. WESZKA, Joan S; ROSENFELD, Azriel. Threshold evaluation techniques. *IEEE Transactions on systems, man, and cybernetics*. 1978, roč. 8, č. 8, s. 622–629.

11. ILLINGWORTH, John; KITTLER, Josef. A survey of the Hough transform. *Computer vision, graphics, and image processing*. 1988, roč. 44, č. 1, s. 87–116.

12. LEAVERS, VF. Which hough transform? *CVGIP: Image understanding*. 1993, roč. 58, č. 2, s. 250–264.

13. ILLINGWORTH, J.; KITTLER, J. The Adaptive Hough Transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1987, roč. PAMI-9, č. 5, s. 690–698. Dostupné z DOI: 10.1109/TPAMI.1987.4767964.

14. JENSEN, Jeppe. Hough transform for straight lines. *Miniproject in Image Processing*. 2007.

15. FARAG, Wael; SALEH, Zakaria. Road lane-lines detection in real-time for advanced driving assistance systems. In: *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. 2018, s. 1–8.

16. FARAG, W.; SALEH, Z. Road Lane-Lines Detection in Real-Time for Advanced Driving Assistance Systems. In: *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. 2018, s. 1–8. Dostupné z DOI: 10.1109/3ICT.2018.8855797.

17. ANTHONY, Martin; BARTLETT, Peter L. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

18. ABDI, Herve. A neural network primer. *Journal of Biological Systems*. 1994, roč. 2, č. 03, s. 247–281.

19. KVASNIČKA, Vladimıér; BEŇUŠKOVÁ, L'ubica; POSPIÉCHAL, Jiřıé; FARKAŠ, Igor; TIŇO, Peter; KRÁL', Andrej. Úvod do teórie neurónových sietıé. In: 1997.

20. SINČÁK, Peter; ANDREJKOVÁ, Gabriela. Neurónové siete Inžiniersky prıéstup (1. diel). *Elfa: Kosice*. 1996.

21. KARN, Ujjwal. An intuitive explanation of convolutional neural networks. *The data science blog*. 2016.

22. FUKUSHIMA, Kunihiko. Neocognitron for handwritten digit recognition. *Neurocomputing*. 2003, roč. 51, s. 161–180.

23. NGIAM, Jiquan; CHEN, Zhenghao; CHIA, Daniel; KOH, Pang; LE, Quoc; NG, Andrew. Tiled convolutional neural networks. *Advances in neural information processing systems*. 2010, roč. 23, s. 1279–1287.

24. LE, Quoc V et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. *Google Brain*. 2015, s. 1–20.

25. KARPATHY, Andrej et al. Cs231n convolutional neural networks for visual recognition. *Neural networks*. 2016, roč. 1, č. 1.

26. YAMASHITA, Rikiya; NISHIO, Mizuho; DO, Richard Kinh Gian; TOGASHI, Kaori. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*. 2018, roč. 9, č. 4, s. 611–629.

27. LAWRENCE, Steve; GILES, C Lee; TSOI, Ah Chung; BACK, Andrew D. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*. 1997, roč. 8, č. 1, s. 98–113.

28. ALBAWI, Saad; MOHAMMED, Tareq Abed; AL-ZAWI, Saad. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, s. 1–6.

29. TRAORE, Boukaye Boubacar; KAMSU-FOGUEM, Bernard; TANGARA, Fana. Deep convolution neural network for image recognition. *Ecological Informatics*. 2018, roč. 48, s. 257–268.

30. LECUN, Yann; RANZATO, M. Deep learning tutorial. In: *Tutorials in International Conference on Machine Learning (ICML'13)*. 2013, s. 1–29.

31. WANG, Ze; REN, Weiqiang; QIU, Qiang. Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint arXiv:1807.01726*. 2018.

32. GEIGER, Andreas; LAUER, Martin; WOJEK, Christian; STILLER, Christoph; URTASUN, Raquel. 3d traffic scene understanding from movable platforms. *IEEE transactions on pattern analysis and machine intelligence*. 2013, roč. 36, č. 5, s. 1012–1025.

33. BOSAGHZADEH, Alireza; ROUTEH, Seidfarbod Seidali. A novel PCA perspective mapping for robust lane detection in urban streets. In: *2017 Artificial Intelligence and Signal Processing Conference (AISP)*. 2017, s. 145–150.

34. ZHENG, Banggui; TIAN, Bingxiang; DUAN, Jianmin; GAO, Dezhi. Automatic detection technique of preceding lane and vehicle. In: *2008 IEEE International Conference on Automation and Logistics*. 2008, s. 1370–1375.

35. HASELHOFF, Anselm; KUMMERT, Anton. 2D line filters for vision-based lane detection and tracking. In: *2009 International Workshop on Multidimensional (nD) Systems*. 2009, s. 1–5.

36. SON, Jongin; YOO, Hunjae; KIM, Sanghoon; SOHN, Kwanghoon. Real-time illumination invariant lane detection for lane departure warning system. *Expert Systems with Applications*. 2015, roč. 42, č. 4, s. 1816–1824.

37. AMINI, Hojatolah; KARASFI, Babak. New approach to road detection in challenging outdoor environment for autonomous vehicle. In: *2016 Artificial Intelligence and Robotics (IRANOPEN)*. 2016, s. 7–11.

38. KONG, Hui; SARMA, Sanjay E; TANG, Feng. Generalizing Laplacian of Gaussian filters for vanishing-point detection. *IEEE Transactions on Intelligent Transportation Systems*. 2012, roč. 14, č. 1, s. 408–418.

39. HERVIEU, Alexandre; SOHEILIAN, Bahman. Road side detection and reconstruction using LIDAR sensor. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, s. 1247–1252.

Dodatok A

Prílohy

V priloženom archíve sa nachádza nasledujúci zoznam zložiek a súborov, ktoré môžu byť využite pri testovaní algoritmov v tejto diplomovej práci.

- **prilohy/code/draw_detected_lanes.py** - hlavný zdrojový kód na detekciu čiar z videa.
- **prilohy/code/full_CNN_model.h5** - model neurónovej siete.
- **prilohy/code/fully_conv_NN.py** - súbor s funkciami pre trénovanie neurónovej siete.
- **prilohy/code/line_detector.py** - zdrojový kód s funkciami pre detekciu čiar po prevedení ROI.
- **prilohy/code/loader.py** - zdrojový kód pre načítavanie trenovacích sád
- **prilohy/code/test.py** - zdrojový kód na testovanie obrázkov na datasete s možnosťou postupného ukladania.
- **prilohy/dataset/Testovanie/540p** - priečinok s testovacími obrázkami a anotovanými dátami v rozlíšení 540p
- **prilohy/dataset/Testovanie/1080p** - priečinok s testovacími obrázkami a anotovanými dátami v rozlíšení 1080p.
- **prilohy/dataset/Trenovanie** - priečinok s trénovacími dátami a anotáciami.
- **prilohy/test.mp4** - výstupné video programu

Práca využíva hlavne knič nice Kesras 2.3.1, OpenCV 4.4.0.46, Tensorflow 2.2.0. Taktiež sú používané knižnice numpy 1.18.5, moviepy 1.0.3, pickleshare 0.7.5, scikit-learn 0.24.1.